WAVELET DECOMPOSITION AND RECONSTRUCTION OF IMAGES USING TRANSPUTER ARRAYS

A Thesis submitted

in partial fulfillment of the requirements

for the Degree of

MASTER OF TECHNOLOGY

by

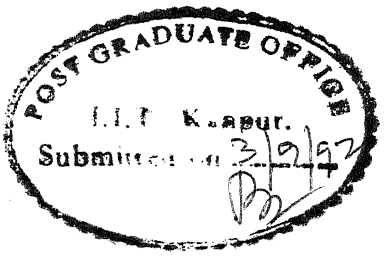
ANURAG SRIVASTAVA

to the

DEPARTMENT OF ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY KANPUR

AUGUST 1992



CERTIFICATE

This is to certify that the thesis work entitled "WAVELET DECOMPOSITION AND RECONSTRUCTION OF IMAGES USING TRANSPUTER ARRAYS" has been carried out by Anurag Srivastava under my supervision and has not been submitted elsewhere for a degree.

Dr. Sumana Gupta
Assistant Professor
Department Of Electrical Engineering
Indian Institute Of Technology
KANPUR

ABSTRACT

In this thesis, an attempt has been made to study and implement the concepts of Wavelet Transforms and multiresolution approximation of images on a network of Transputers. The decomposition and reconstruction algorithms are derived first for one–dimensional case and then extended to the two–dimensional case,i.e.,the case of images. These algorithms are implemented on a linear array of eight transputers using the data–partitioning approach. The performance of Wavelet decomposition and reconstruction on noise–corrupted images are also studied. All the program coding is done in Occam, the native language for the transputers. The program development, testing and debugging is done under the Transputer Development System environment. The performance of the various basis functions used are compared.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to *Dr. Sumana Gupta*, for her ever available counsel and guidance. She has been a constant source of inspiration and motivation during all phases of my interaction with her. Words are too few to describe the feeling of gratitude for the immense co–operation offered by her, not only in this work but in all other ways as well.

I am thankful to *Prof. M. U. Siddiqi* for allowing me to use the IP Lab facilities with the maximum possible freedom.

I will always be indebted to *Alok, Udaya, Galgali, Subbarao and Madhu* for their help and co–operation during this work.

I thank my friends *Hemant, Aloo, Tamatar, Peeyush, SSG, Kholu, Amitabh, Chanchal, Vivek, Vandana and others* for making my stay here memorable and also for helping me prepare for the *mother of all exams.*

Finally, I would like to thank my parents who have made me whatever I am today. Needless to add, there's is a debt which I will never be able to repay, no matter how much I try.

I can not thank *Saloni*. There's no concept of thanks in love.

Anurag

Dedicated to

all my near and dear ones,

past, present and future.

TABLE OF CONTENTS

CHAPTER 3 TRANSPUTER AND OCCAM

LIST OF PLATES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 WAVELETS AND SIGNAL PROCESSING

Wavelet theory provides a unified framework for a number of techniques which had been developed independently for various signal processing applications. For example, multiresolution signal processing, used in computer vision; sub band coding, developed for speech and image compression; and wavelet series expansions, developed in applied mathematics have been recently recognized as different views of a single theory.

Wavelet theory covers quite a large area. It treats both the continuous and discrete time cases. It provides very general techniques that can be applied to many tasks in signal processing and therefore has many potential applications.

In particular, the Wavelet transform (WT) is of interest for the analysis of non–stationary signals, because it provides an alternative to the classical Short–time Fourier Transform (STFT) or Gabor Transform (GT). The basic difference is as follows. In contrast to the STFT, which uses a single analysis window, the WT uses short windows at high frequencies and long windows at low frequencies. This is in accordance to the so–called "constant–Q" or constant relative bandwidth frequency analysis.

For some applications it is desirable to see the WT as a signal decomposition onto a set of basis functions. In fact, basis functions called wavelets always underlie the wavelet analysis. They are obtained from a single prototype wavelet by dilations and contractions (scaling) as well as translations (shifts). The

prototype wavelet can be thought of as a band pass filter, and the constant–Q property of the other band pass filters (wavelets) follows because they are scaled versions of the prototype.

Therefore, in a WT, the notion of scale is introduced as an alternative to frequency, leading to a so–called time–scale representation. This means that a signal is mapped into a time–scale plane (the equivalent of the time–frequency plane used in the STFT).

There are several types of wavelet transforms, and depending on the application one may be preferred to the others. For a continuous input signal, the time and scale parameters can be continuous [4], leading to the Continuous Wavelet Transform (CWT). They may as well be discrete [5, 3, 6], leading to a Discrete Wavelet Transform (DWT). In the latter case it uses multirate signal processing techniques [7] and is related to sub band coding schemes used in speech and image compression.

Wavelet theory has been developed as a unifying framework only recently, although similar ideas and constructions took place as early as the beginning of the century e.g., the works of P.Franklin(1928), A.Haar(1910), J.Littlewood and R.Paley(1937), A.Calderon(1964) etc. The idea of looking at a signal at various scales and analyzing it with various resolutions has in fact emerged independently in many different fields of mathematics, physics and engineering. In the late eighties, Daubechies and Mallat, in addition to their contribution to the theory of wavelets, established connections to discrete signal processing results [5, 1]. Since then, a number of theoretical as well as practical contributions have been made on various aspects of WT's, and the subject is growing rapidly [8].

1.2 MULTIRESOLUTION SIGNAL DECOMPOSITION

In computer vision, it is difficult to analyze the information content of an image directly from the gray–level intensity of the image pixels. Indeed, this value depends on the lighting conditions. More important are the local variations of the image intensity. The size of the neighborhood where the contrast is computed must be adapted to the size of the objects that we want to analyze [9]. This size defines a resolution of reference for measuring the local variations of the image. Generally the structures we want to recognize have very different sizes. Hence, it is not possible to define a priori an optimal resolution for analyzing images. Several researchers for example, E.Hall, J.Rouge, R.Wong, D.Marr, T.Poggio, A.Rosenfeld and M.Thurston etc. have developed pattern matching algorithms which process the image at different resolutions. For this purpose, one can reorganize the image information into a set of details appearing at different resolutions . Given a sequence of increasing resolutions $(r_j)_{j\epsilon Z}$, the details of an image at the resolution r_j are defined as the difference of information between its approximation at resolution r_j and its approximation at the resolution r_{j-1} .

A multiresolution decomposition helps us to have a scale–invariant interpretation of the image. The scale of an image depends upon the distance between the scene and the optical center of the camera. When the image scale is modified, our interpretation of the scene should not change. A multiresolution representation can be partially scale–invariant if the sequence of resolution parameters $(r_j)_{j\epsilon Z}$ varies exponentially. Assume that there exists a resolution step $a \epsilon R$ such that for all integers j ; $r_j = a^j$. If the camera gets a times closer to the scene, each object of the scene is projected onto an area a^2 times bigger in the focal plane of the camera. That is, each object is measured at a resolution a times bigger. Hence the details of this new image at the resolution corresponds to

the details of the previous image at the resolution α^{j+1}. Rescaling the image by α translates the image details along the resolution axis. If the image details are processed identically at all resolutions, our interpretation of the image information is not modified.

A multiresolution representation provides a simple hierarchical framework for interpreting the image information as shown by the work of J.Koenderink. At different resolutions, the details of the image generally characterize different physical structures of the scene. At a coarse resolution, these details corresponds to the larger structures which provide the image "context". It is therefore natural to analyze first the image details at a coarse resolution and then gradually increase the resolution. Such a coarse–to–fine strategy is useful for pattern recognition algorithms.

Burt [13] and Cowley have each introduced pyramidal implementations for computing the signal details at different resolutions. In order to simplify the computations, Burt has chosen a resolution step α equal to 2. The details at each resolution 2^j are calculated by filtering the original image with the difference of two low–pass filters and by sub sampling the resulting image by a factor 2^j. This operation is performed over a finite range of resolutions. In this implementation, the difference of two low–pass filters gives an approximation of the Laplacian of the Gaussian. The details at different resolutions are regrouped into a pyramid structure called the Laplacian pyramid. However, the Laplacian pyramid structures as studied by Burt and Cowley, suffer from the difficulty that data at several levels are correlated. There is no clear model which handles this correlation. It is thus difficult to know whether a similarity between the image details at different resolutions is a characteristic of the image itself or to the intrinsic redundancy of the representation. Furthermore, the Laplacian multiresolution representation does

not introduce any spatial orientation selectivity into the decomposition process. This spatial homogeneity can be inconvenient for pattern recognition problems such as texture discrimination as shown by the work of B.Julesz.

Here, we have transformed a function into an approximation at a resolution 2^j. The difference of information between two approximations at the resolutions 2^{j+1} and 2^j is extracted by decomposing the function in a wavelet orthonormal basis. This decomposition defines a compete and orthogonal multiresolution representation called the wavelet representation. Wavelets have been introduced by Grossman and Morlet as functions $\Psi(x)$ whose translations and dilations $(\sqrt{s}\,\Psi(sx-t))_{(s,t)} \in R^+ \times R$ can be used for expansions of $L^2(R)$ functions. Meyer [35] showed that there exists wavelets $\Psi(x)$ such that $(\sqrt{2^j}\Psi(2^jx-t)_{(j,k)\in Z^2}$ is an orthonormal basis of $L^2(R)$. These bases generalize the Haar basis. The wavelet orthonormal bases provide an important new tool in functional analysis. Indeed, before them it had been believed that no construction could yield simple orthonormal bases of $L^2(R)$ whose elements had good localization properties in both the spatial and Fourier domains.

The multiresolution approach to wavelets enables us to characterize the class of functions $\Psi(x) \in L^2(R)$ that generate an orthonormal basis. The model is first described for one–dimensional signals and then extended to two dimensions for image processing. The wavelet representation of images discriminates several spatial orientations. The computation of the wavelet representation may be accomplished with a pyramidal algorithm based on convolutions with quadrature mirror filters. The signal can also be reconstructed from a wavelet representation with a similar pyramidal algorithm.

1.3 THE ALGORITHMS

A_{2^j} is the operator which approximates a signal at a resolution 2^j. Since computers can only process discrete signals, we must work with discrete approximations. Thus $A^d_{2^j}f$ is called a *discrete approximation* of f(x) at the resolution 2^j. $D_{2^j}f$ is called the *discrete detail signal* at the resolution 2^j. It contains the difference of information between $A^d_{2^{j+1}}f$ and $A^d_{2^j}f$. The original discrete signal $A^d_1 f$ measured at the resolution 1 is represented by $(A^d_{2^{-J}}f , (D_{2^j}f)_{-J \le j \le -1})$.

This set of discrete signals is called an orthogonal wavelet representation,and consists of the reference signal at a coarse resolution $A^d_{2^{-J}}f$ and the detail signals at the resolution 2^j for $-J \le j \le -1$ .It can be interpreted as a decomposition of the original signal in an orthonormal wavelet basis. $\Psi(x)$ is the orthogonal wavelet and the wavelet orthonormal basis is built by scaling and translating the function $\Psi(x)$. $\phi(x)$ is the scaling function and H the corresponding conjugate filter. The impulse response of the filter G is related to the impulse response of the filter H by

$$g(n) = (-1)^{1-n} h(1-n).$$

G is the mirror filter of H, and is a high pass filter.In signal processing, G and H are called *quadrature mirror filters.*

The wavelet model can be generalized to any dimension $n > 0$. We have specifically studied the two–dimensional case for image processing applications.The signal is now a finite energy function $f(x,y) \in L^2(R^2)$.A multiresolution approximation of $L^2(R^2)$ is a sequence of subspaces of $L^2(R^2)$ which satisfies a straightforward two–dimensional extension.

The Decomposition Algorithm : The two dimensional wavelet transform can be seen as a one–dimensional wavelet transform along the x and y axes. It can be computed with a separable extension of the one–dimensional decomposition algorithm. At each step we decompose $A^d_{2^{j+1}}f$ into $A^d_{2^j}f$, $D^1_{2^j}f$,$D^2_{2^j}f$, and $D^3_{2^j}f$. This algorithm is illustrated by the block diagram in Figure 2.1. We first convolve the rows of $A^d_{2^{j+1}}f$ with a one–dimensional filter, retain every other row, convolve the columns of the resulting signals with another one–dimensional filter and retain every other column. The filters used in this decomposition are the quadrature mirror filters $\tilde{H}$ and $\tilde{G}$ which are related to the filters H and G as follows:–

$$\tilde{g}(n) = g(-n) \quad \text{and} \quad \tilde{h}(n) = h(-n).$$

The structure of application of the filters is given in the Fig. 2.1. We compute the wavelet transform of an image A^d_1f by repeating this process for $-1 \geq j \geq -J$. This corresponds to a separable conjugate mirror filter decomposition.

The Reconstruction Algorithm : The one–dimensional reconstruction algorithm can also be extended to two dimensions. At each step, the image $A^d_{2^{j+1}}f$ is reconstructed from $A^d_{2^j}f$,$D^1_{2^j}f$,$D^2_{2^j}f$,and $D^3_{2^j}f$. This algorithm is illustrated in Figure 2.2. Between each column of the coarse and detail images we add a column of zeros, convolve the rows with one–dimensional filter, add a row of zeros between each row of the resulting image, and convolve the columns with another one–dimensional filter. The filters used in reconstruction are the quadrature mirror filters H and G. The image A^d_1f is reconstructed from its wavelet transform by repeating this process for $-J \leq j \leq -1$. Figure 2.2 shows the reconstruction of the original image from its wavelet representation.

1.4 TRANSPUTERS AND OCCAM

The word *Transputer* may be interpreted as a contraction of the word *Tranceiver* and *computer*. The best known Transputers are the IMS T212, T41 and T800. The T212 is a 16–bit processor, the other two are 32–bit processors The T800 has a full IEEE floating–point processor on chip.Communication betwee the processes is achieved by means of channels. *Occam* is the native language fo Transputers.To gain the most benefit from the Transputer architecture, the whol system can be programmed in Occam. This provides all the advantages of a hig level language, the maximum program efficiency and the ability to use the specia features of the Transputer.The pattern in which the processors are connecte together is known as ***topology*** or the configuration. Designing a topology for large system can be difficult. Transputer software is mostly developed under th '*Transputer development environment*' (TDS) supplied by INMOS.

1.5.IMPLEMENTATION OF THE ALGORITHMS ON A TRANSPUTER NETWORK

The above decomposition and reconstruction algorithms have beer implemented using transputers. In any image processing application, the time taker by the procedure is of crucial importance, especially so in real time applications The concept of parallel processing comes in handy here. Our equipment consists o a root processor and 8 other processors. The root processor and 4 other processors are of the T800 type while the remaining 4 are of the T414 type

The configuration being considered is a linear array of the processors with data flow along a linear path. We have utilized the concept of data–partitioning or in parallel processing parlance, "*coarse grain concurrency*".

1.6 ORGANIZATION OF THE THESIS

The Thesis has been organized as follows:

Chapter 1 is a brief introduction to the entire work. Its purpose is to give an overview of the theoretical as well as the implementation aspects of the topic. It covers the role of wavelets in signal processing, the concept of multiresolution transforms, the decomposition and reconstruction algorithms, an introduction to Transputers and its language Occam.

Chapter 2 gives a brief survey of the theoretical aspects related to wavelet representation and. It starts with an introduction to the notation used throughout the thesis work. It then dwells on the multiresolution approximation of $L^2(R)$ and also enunciates the properties of A_{2^j} which is the operator approximating the signal at a resolution 2^j. The wavelet representation and its implementation is then discussed. Next, the theoretical basis of signal reconstruction from an orthogonal representation is studied. All the previous concepts are then extended to the domain of images i.e., 2–Dimensions. The decomposition and reconstruction algorithms in case of images are discussed.

Chapter 3 gives a brief insight into Transputers, the machines used for implementing the algorithms. It introduces the Transputers and then gives the architectural details like the instruction processor, the instruction set, the memory controller, the process scheduler, internal and external communication, the communication links, the timer and floating point processor. Occam is then introduced and a brief review of programming transputers is done. It also gives the hardware setup of various network configurations including the actual configuration which has been used to implement the decomposition and reconstruction algorithms.

Chapter 4 discusses the actual data transfer mechanism. It also gives th details of the image used, the choice of basis functions, the introduction of nois etc.

Chapter 5 concludes the Thesis with a brief discussion on the result achieved, the conclusions drawn, and some suggestions for further work.

Appendix describes the way to derive the filter coefficients from the basi functions.

CHAPTER 2

MULTIRESOLUTION TRANSFORM

2.1 NOTATION.

Z and R denote the set of integers and real numbers respectively. $L^2(R)$ denotes the vector space of measurable, square–integrable one–dimensional functions f(x). For $f(x) \in L^2(R)$ and $g(x) \in L^2(R)$, the inner product of f(x) with g(x) is written

$$< g(u),\ f(u) > = \int_{-\infty}^{+\infty} g(u)\ f(u)\ du$$

The norm of f(x) in $L^2(R)$ is given by

$$\| f \|^2 = \int_{-\infty}^{+\infty} |f(u)|^2\ du$$

The convolution of two functions $f(x) \in L^2(R)$ and $g(x) \in L^2(R)$ is denoted by

$$f * g(x) = (f(u) * g(u))\ (x)$$

$$= \int_{-\infty}^{+\infty} g(x-u)\ f(u)\ du$$

The Fourier transform of $f(x) \in L^2(R)$ is written $\hat{f}(\omega)$ and is defined by

$$\hat{f}(\omega) = \int_{-\infty}^{+\infty} f(x)\ e^{-jwx}\ dx.$$

$I^2(Z)$ is the vector space of square–summable sequences.

$$I^2(Z) = \{\ (\alpha_i)_i \in Z : \sum_{-\infty}^{\pm\infty} |\ \alpha_i|^2 < \infty\ \}.$$

$L^2(R^2)$ is the vector space of measurable, square–integrable two–dimensional functions f(x,y). For $f(x,y) \in L^2(R^2)$ and $g(x,y) \in L^2(R^2)$, the inner product of f(x,y) with g(x,y) is written

$$< f(x,y),\ g(x,y) > = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} f(x,y)\ g(x,y)\ dx\ dy$$

The Fourier Transform of $f(x,y) \in L^2(R^2)$ is written $\hat{f}(\omega_x,\omega_y)$ and is defined by

$$\hat{f}(\omega_x,\omega_y) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} f(x,y)\ e^{-j(\omega_x x\ +\ \omega_y y)}\ dx\ dy$$

2.2 MULTIRESOLUTION APPROXIMATION OF $L^2(R)$

Let A_{2^j} be the operator which approximates the signal at a resolution 2^j. We suppose that our original signal f(x) is measurable and has a finite energy : $f(x) \in L^2(R)$. We characterize A_{2^j} from the properties one would expect from such an approximation operator.

(1). A_{2^j} is a linear operator. If $A_{2^j} f(x)$ is the approximation of some function f(x) at the resolution 2^j, then $A_{2^j} f(x)$ is not modified if we approximate it again at the resolution 2^j. This principle shows that $A_{2^j} \cdot A_{2^j} = A_{2^j}$ .The operator A_{2^j} is thus a projection operator on a particular vector space $V_{2^j} \subset L^2(R)$. The vector space V_{2^j} can be interpreted as the set of all possible approximations at the resolution 2^j of functions in $L^2(R)$.

(2). Among all approximated functions at the resolution 2^j, $A_{2^j} f(x)$ is the function which is most similar to f(x).

$$\forall\ g(x) \in V_{2^j}\ ,\ \| g(x)-f(x) \| \geq \| A_{2^j} f(x) - f(x) \| \qquad (1)$$

Hence the operator A_{2^j} is an orthogonal projection on the vector space V_{2^j}.

(3). The approximation of a signal at a resolution 2^{j+1} contains all the necessary information to compute the same signal at a smaller resolution 2^j. This is a causality property. Since A_{2^j} is a projection operator on V_{2^j} this principle is equivalent to

$$\forall\ j \in Z\ , V_{2^j} \subset V_{2^{j+1}}. \qquad (2)$$

(4). An approximation operation is similar at all resolutions. The spaces of approximated functions should thus be derived from one another by scaling each approximated function by the ratio of their resolution values

$$\forall\ j \in Z\ ,\quad f(x) \in V_{2^j} \Leftrightarrow f(2x) \in V_{2^{j+1}}. \tag{3}$$

(5). The approximation $A_{2^j}\ f(x)$ of a signal $f(x)$ can be characterized by 2^j samples per length unit. When $f(x)$ is translated by a length proportional to 2^{-j}, $A_{2^j}\ f(x)$ is translated by the same amount and is characterized by the same samples which have been translated . As a consequence of (3), it is sufficient to express the principle (5) for the resolution $j=0$. The mathematical translations consists of the following.

Discrete Characterization:

$$\text{There exists an isomorphism I from } V_1 \text{ onto } l^2(Z). \tag{4}$$

Translation of the approximation:

$$\forall\ k \in Z\ ,\ A_1 f_k(x) = A_1 f(x-k)\ ,\ \text{where } f_k(x) = f(x-k). \tag{5}$$

Translation of the samples :

$$I(A_1\ f(x)) = (\alpha_i)_{\ i\in Z} \Leftrightarrow I(A_1\ f_k(x)) = (\alpha_{i-k})_{i,k\in Z} \tag{6}$$

(6). When computing an approximation of $f(x)$ at resolution 2^j, some information about $f(x)$ is lost. However, as the resolution increases to $+\infty$ the approximated signal should converge to the original signal. Conversely as the resolution decreases to zero, the approximated signal contains less and less information and converges to zero.

Since the approximated signal at resolution 2^j is equal to the orthogonal

projection on a space V_{2^j} , this principle can be written

$$\lim_{j \to \infty} V_{2^j} = \overline{\bigcup_{-\infty}^{+\infty}} V_{2^j} \text{ is dense in } L^2(R) \tag{7}$$

and

$$\lim_{j \to -\infty} V_{2^j} = \bigcap_{-\infty}^{+\infty} V_{2^j} = \{0\}. \tag{8}$$

We call any set of vector spaces $(V_{2^j})_{j \in Z}$ which satisfies the properties (2)–(8) a multiresolution approximation of $L^2(R)$. The associated set of operators A_{2^j} satisfying (1)–(6) give the approximation of any $L^2(R)$ function at a resolution 2^j.

Theorem 1 : Let $(V_{2^j})_{j \in Z}$ be a multiresolution approximation of $L^2(R)$. There exists a unique function $\phi(x) \in L^2(R)$, called a scaling function, such that if we set $\phi_{2^j}(x) = 2^j \phi(2^j x)$ for $j \in Z$,(the dilation of $\phi(x)$ by 2^j), then $\sqrt{2^{-j}}\ \phi_{2^j}(x - 2^{-j}n))_{n \in Z}$ is an orthonormal basis of V_{2^j}. (9)

This theorem shows that we can build an orthonormal basis of any V_{2^j} by dilating a function $\phi(x)$ with a coefficient 2^j and translating the resulting function on a grid whose interval is proportional to 2^{-j}. The function $\phi_{2^j}(x)$ are normalized with respect to the $L^1(R)$ norm. The coefficient $\sqrt{2^{-j}}$ appears in the basis set in order to normalize the functions in $L^2(R)$ norm. For a given multiresolution approximation $(V_{2^j})_{j \in Z}$, there exists a unique scaling function which satisfies (9). However, for different multiresolution approximations, the scaling functions are different. In general, we want to have a smooth scaling function. The Fourier Transform of a continuously differentiable and exponentially decreasing scaling function has the shape of a low–pass filter.

The orthogonal projection on V_{2^j} can now be computed by decomposing the signal f(x) on the orthonormal basis given by Theorem 1. The approximation of the signal f(x) at the resolution 2^j, $A_{2^j}f$ is thus characterized by the set of inner products which we denote by

$$A^d_{2^j}f = (< f(u) , \phi_{2^j}(u - 2^{-j}n) >)_{n \in Z} \qquad 10)$$

$A^d_{2^j}f$ is called the discrete approximation of f(x) at the resolution 2^j. Since computers can only process discrete signals, we must work with discrete approximations. Each inner product can also be interpreted as a convolution product evaluated at a point $2^{-j}n$

$$< f(u) , \phi_{2^j}(u - 2^{-j}n) > = \int_{-\infty}^{+\infty} f(u)\ \phi_{2^j}(u - 2^{-j}n)\ du = f(u) * \phi_{2^j}(-u))\ (2^{-j}n)$$

Hence we can rewrite $A^d_{2^j}f$

$$A^d_{2^j}f = ((\ f(u) * \phi_{2^j}(-u))\ (\ 2^{-j}n))_{n \in Z} \qquad (11)$$

Since $\phi(x)$ is a low–pass filter, this discrete signal can be interpreted as a low–pass filtering of f(x) followed by a uniform sampling at the rate 2^j. In an approximation operation, when removing the details of f(x) smaller than 2^{-j}, we suppress the highest frequencies of this function. The scaling function $\phi(x)$ forms a very particular low–pass filter since the family of functions $(\sqrt{2^{-j}}\ \phi_{2^j}(x - 2^{-j}n))_{n \in Z}$ is an orthonormal family.

The discrete approximation of f(x) at the resolution 2^j can be computed with a pyramidal algorithm.

2.3 IMPLEMENTATION OF A MULTIRESOLUTION TRANSFORM

In practice, a physical measuring device can only measure a signal at a finite resolution. For normalization purposes, we suppose that this resolution is equal to 1. Let $A_1^d f$ be the discrete approximation at the resolution 1 that is measured. The causality principle says that from $A_1^d f$ we can compute all the discrete approximations $A_{2^j}^d f$ for $j < 0$. In this section, we describe a simple iterative algorithm for calculating these discrete approximations.

Let $(V_{2^j})_{j \in Z}$ be a multiresolution approximation and $\phi(x)$ be the corresponding scaling function. The family of functions $(\sqrt{2^{-j-1}}\ \phi_{2^j+1}(x - 2^{-j-1}k))_{k \in Z}$ is an orthonormal basis of V_{2^j+1}. We know that for any $n \in Z$, the function $\phi_{2^j}(x - 2^{-j}n)$ is a member of V_{2^j} which is included in V_{2^j+1}. It can thus be expanded in this orthonormal basis of V_{2^j+1} :

$$\begin{aligned}\phi_{2^j}(x - 2^{-j}n) &= \\ 2^{-j-1} \sum_{-\infty}^{+\infty} &< \phi_{2^j}(u - 2^{-j}n)\ ,\ \phi_{2^j+1}(u - 2^{-j-1}k) > \\ &\times \phi_{2^j+1}(x - 2^{-j-1}k)) \end{aligned} \tag{12}$$

By changing variables in the inner product integral, one can show that

$$\begin{aligned} 2^{-j-1} &< \phi_{2^j}(u - 2^{-j}n)\ ,\ \phi_{2^j+1}(u - 2^{-j-1}k) > \\ &= < \phi_{2^{-1}}(u)\ ,\ \phi(u - (k - 2n)) > \end{aligned}$$

When computing the inner products of f(x) with both sides of (12) we obtain

$$\begin{aligned} &< f(u)\ ,\ \phi_{2^j}(u - 2^{-j}n) > \\ &= \sum_{-\infty}^{+\infty} < \phi_{2^{-1}}(u)\ ,\ \phi(u - (k - 2n)) > \\ &\times < f(u)\ ,\ \phi_{2^j+1}(u - 2^{-j-1}k) > \end{aligned}$$

Let H be the discrete filter whose impulse response is given by

$$\forall\ n \in Z\ ,\ h(n) = < \phi_{2^{-1}}(u)\ ,\ \phi(u - n) > \qquad (14)$$

Let $\tilde{H}$ be the mirror filter with impulse response $\tilde{h}(n) = h(-n)$. By inserting (14) in the previous equation, we obtain

$$< f(u)\ ,\ \phi_{2^j}(u - 2^{-j}n) >$$
$$= \sum_{-\infty}^{+\infty} \tilde{h}(2n - k) < f(u)\ ,\ \phi_{2^{j+1}}(u - 2^{-j-1}k) > \qquad (15)$$

Equation (15) shows that $A^d_{2^j}f$ can be computed by convolving $A^d_{2^{j+1}}f$ with $\tilde{H}$ and keeping every other sample of the output. All the discrete approximations $A^d_{2^{j+1}}f$, for $j< 0$, can thus be computed from $A^d_1 f$ by repeating this process. This operation is called a pyramid transform.

In practice, the measuring device gives only a finite number of samples: $A^d_1 f = (\alpha_n)_{1 \le n \le N}$. Each discrete signal $A^d_{2^j}f(j < 0)$ has $2^j n$ samples. In order to avoid border problems while computing the discrete approximations $A^d_{2^j}f$, we suppose that the original signal $A^d_1 f$ is symmetric with respect to $n = 0$ and $n = N$.

$$\alpha_n = \alpha_{-n} \text{ if } -N < n < 0$$
$$= \alpha_{2N-n} \text{ if } 0 < n < N$$

Theorem 1 shows that a multiresolution approximation $(V_{2^j})_{j \in Z}$ is completely characterized by the scaling function $\phi(x)$. A scaling function can be defined as a function $\phi(x) \in L^2(R)$ such that, for all $j \in Z$, $(\sqrt{2^{-j}}\ \phi_{2^j}(x - 2^{-j}n))_{n \in Z}$ is an orthonormal family, and if V_{2^j} is the vector space generated by this family of functions, then $(V_{2^j})_{j \in Z}$ is a multiresolution approximation of $L^2(R)$. We also impose a regularity condition on the scaling functions. A scaling function

$\phi(x)$ must be continuously differentiable and the asymptotic decay of $\phi(x)$ and $\phi'(x)$ at infinity must satisfy

$$|\phi(x)| = O(x^{-2}) \text{ and } |\phi'(x)| = O(x^{-2}).$$

The following theorem gives a practical characterization of the Fourier Transform of a scaling function.

Theorem 2 : Let $\phi(x)$ be a scaling function, and let H be a discrete filter with impulse response $h(n) = < \phi_{2^{-1}}(u) , \phi(u - n) >$. Let $H(\omega)$ be the Fourier series defined by

$$H(\omega) = \sum_{-\infty}^{+\infty} h(n) \, e^{-jwn} \tag{16}$$

$H(\omega)$ satisfies the following two properties :

$$|H(0)| = 1 \text{ and } h(n) = O(n^{-2}) \text{ at infinity.} \tag{16a}$$

$$|H(\omega)|^2 + |H(\omega + \pi)|^2 = 1 \tag{16b}$$

Conversely let $H(\omega)$ be a Fourier series satisfying (16a) and (16b) and such that

$$|H(\omega)| \neq 0 \quad \text{for} \quad \omega \in [0 , \pi/2]. \tag{16c}$$

The function defined by

$$\hat{\phi}(\omega) = \prod_{p=1}^{+\infty} H(2^{-p}\omega) \tag{17}$$

is the Fourier transform of a scaling function.

The filters that satisfy property (16b) are called *conjugate–filters.* Given a conjugate filter H which satisfies (16a)–(16c), we can then compute the Fourier Transform of the corresponding scaling function with (15). It is possible to chose $H(\omega)$in order to obtain a scaling function $\phi(x)$ which has good localization

properties in both the frequency and spatial domains. The smoothness class of $\phi(x)$ and its asymptotic decay at infinity can be estimated from the properties of $H(\omega)$.

2.4 THE WAVELET REPRESENTATION

As explained in the introduction, we wish to build a multiresolution representation based on the differences of information available at two successive resolutions 2^j and 2^{j+1}. Such a representation can be computed by decomposing the signal using a wavelet orthonormal basis.

The Detail Signal. The difference of information between the approximation of a function $f(x)$ at the resolutions 2^j and 2^{j+1}. is called the detail signal at the resolution 2^j. The approximation of the signal at the resolution 2^j and 2^{j+1} are respectively equal to its orthogonal projection on V_{2^j} and $V_{2^{j+1}}$. By applying the projection theorem, we can easily show that the detail signal at the resolution 2^j is given by the orthogonal projection of the original signal on the orthogonal complement of V_{2^j} in $V_{2^{j+1}}$. Let O_{2^j} be this orthogonal complement, i.e.,

$$O_{2^j} \text{ is orthogonal to } V_{2^j}$$

$$O_{2^j} \oplus V_{2^j} = V_{2^{j+1}}.$$

To compute the orthogonal projection of a function $f(x)$ on O_{2^j}, we need to find an orthogonal basis of O_{2^j}. Much like Theorem 1, Theorem 3 shows that such a basis can be built by scaling and translating a function $\Psi(x)$.

Theorem 3 : Let $(V_{2^j})_{j \in Z}$ be a multiresolution vector space sequence, $\phi(x)$ the scaling function, and H the corresponding conjugate filter. Let $\Psi(x)$ be a function whose Fourier Transform is given by

$$\hat{\Psi}(\omega) = G(\omega/2)\ \hat{\phi}(\omega/2)$$

$$\text{with } G(\omega) = e^{-i\omega}\ \overline{H(\omega + \pi)} \tag{18}$$

Let $\Psi_{2^j}(x) = 2^j\ \Psi(2^j x)$ denote the dilation of $\Psi(x)$ by 2^j. Then $(\sqrt{2^{-j}}\ \Psi_{2^j}(x - 2^{-j}n))_{n \in Z}$ is an orthonormal basis of O_{2^j} and

$(\sqrt{2^{-j}}\ \Psi_{2^j}(x - 2^{-j}n))_{(n,j) \in Z^2}$ is an orthonormal basis of $L^2(R)$.

$\Psi(x)$ is called an *orthogonal wavelet.*

An orthonormal basis of O_{2^j} can thus be computed by scaling the wavelet $\Psi(x)$ with a coefficient 2^j and translating it on a grid whose interval is proportional to 2^{-j}. For computing a wavelet, we can define a function $H(\omega)$ which satisfies the conditions (16a)–(16c) of Theorem 2, compute the corresponding scaling function $\phi(x)$ with equation (17) and the wavelet $\Psi(x)$ with (18). Depending upon choice of $H(\omega)$, the scaling function $\phi(x)$ and the wavelet $\Psi(x)$ can have good localization both in the spatial and Fourier domains. Daubechies [9] studied the properties of $\phi(x)$ and $\Psi(x)$ depending upon $H(\omega)$. She shows that for any $n > 0$, we can find a function $H(\omega)$ such that the corresponding wavelet $\Psi(x)$ has a compact support and is n times continuously differentiable.

The decomposition of a signal in an orthonormal wavelet basis gives an intermediate representation between Fourier and spatial representations. Due to this double localization in the Fourier and the spatial domains, it is possible to characterize the local regularity of a function f(x) based on the coefficients in a wavelet orthonormal basis expansion [25].

Let $P_{O_{2j}}$ be the orthogonal projection on the vector space O_{2j}. As a consequence of Theorem 3, this operator can now be written

$$P_{O_{2j}} f(x) = 2^{-j} \sum_{-\infty}^{+\infty} < f(u) , \Psi_{2j}(u - 2^{-j}n) > \times \Psi_{2j}(x - 2^{-j}n). \quad (19)$$

$P_{O_{2j}} f(x)$ yields to the detail signal of $f(x)$ at the resolution 2^j. It is characterized by the set of inner products

$$D_{2j}f = (< f(u) , \Psi_{2j}(u - 2^{-j}n) >)_{n \in Z} \quad (20)$$

$D_{2j}f$ is called the discrete detail signal at the resolution 2^j. It contains the difference of information between $A^d_{2j+1} f$ and $A^d_{2j} f$. As we did in (11), we can prove that each of these inner products is equal to the convolution of $f(x)$ with $\Psi_{2j}(-x)$ evaluated at $2^{-j}n$

$$< f(u) , \Psi_{2j}(u - 2^{-j}n) > = (f(u) * \Psi_{2j}(-u)) (2^{-j}n) \quad (21)$$

Equations (20) and (21) show that the discrete detail signal at the resolution 2^j is equal to a uniform sampling of $(f(u) * \Psi_{2j}(-u))$ (x) at the rate 2^j

$$D_{2j}f = ((f(u) * \Psi_{2j}(-u)) (2^{-j}n))_{n \in Z}$$

The wavelet $\Psi(x)$ can be viewed as a band pass filter whose frequency bands are approximately equal to $[-2\pi, -\pi] \cup [\pi, 2\pi]$. Hence the detail signal $D_{2j}f$ describes $f(x)$ in the frequency bands $[-2^{-j+1}\pi, -2^{-j}\pi] \cup [2^{-j}\pi, 2^{-j+1}\pi]$.

We can prove by induction that for any $J > 0$, the original discrete signal $A^d_1 f$ measured at the resolution 1 is represented by

$$(A^d_{2^{-J}} f , (D_{2j}f)_{-J \leq j \leq -1}. \quad (22)$$

This set of discrete signals is called an *orthogonal wavelet representation*, and consists of the reference signal at a coarse resolution $A^d_{2^{-J}} f$ and the detail signals at the resolutions 2^j for $-J \leq j \leq -1$. It can be interpreted as a decomposition

of the original signal in an orthonormal wavelet basis or as a decomposition of the signal in a set of independent frequency channels. This independence is due to the orthogonality of the wavelet functions.

In analogy with the Laplacian pyramid data structure, $A_2^d{}^{-J} f$ provides the top – level Gaussian pyramid data, and the $D_{2^j} f$ provide the successive Laplacian pyramid levels. Unlike the Laplacian pyramid, however, there is no over sampling, and the individual coefficients in the set of data are independent.

2.5 IMPLEMENTATION OF AN ORTHOGONAL WAVELET REPRESENTATION

Here, we describe a pyramidal algorithm to compute the wavelet representation. We show that $D_{2^j} f$ can be calculated by convolving $A_{2^{j+1}}^d f$ with a discrete filter G whose form we will characterize.

For any $n \in Z$, the function $\Psi_{2^j}(x - 2^{-j}n)$ is a member of $O_{2^j} \subset V_{2^{j+1}}$. In the same manner as (12), this function can be expanded in an orthonormal basis of $V_{2^{j+1}}$.

$$\Psi_{2^j}(x - 2^{-j}n) = 2^{-j-1} \sum_{-\infty}^{+\infty} < \Psi_{2^j}(u - 2^{-j}n) , \phi_{2^{j+1}}(u - 2^{-j-1}k) > \times \phi_{2^{j+1}}(x - 2^{-j-1}k)) \tag{23}$$

As we did in (13), by changing variables in the inner product integral we can prove that

$$2^{-j-1} < \phi_{2^j}(u - 2^{-j}n) , \phi_{2^{j+1}}(u - 2^{-j-1}k) > = < \Psi_{2^{-1}}(u) , \phi(u - (k - 2n)) > \tag{24}$$

Hence, by computing the inner product of f(x) with the functions of both sides of (24), we obtain

$$< f(u) , \Psi_{2^j}(u - 2^{-j}n) > = \sum_{-\infty}^{+\infty} < \Psi_{2^{-1}}(u) , \phi(u - (k - 2n)) > \times < f(u) , \phi_{2^j+1}(u - 2^{-j-1}k) > \quad (25)$$

Let G be the discrete filter whose impulse response is given by

$$\forall\, n \in Z , \; g(n) = < \Psi_{2^{-1}}(u) , \phi(u - n) > \quad (26)$$

and let $\tilde{G}$ be the symmetric filter with impulse response $\tilde{g}(n) = g(-n)$. The transfer function of this filter is the function $G(\omega)$ defined in Theorem 3, equation (18). By inserting (27) in (26), we obtain

$$< f(u) , \Psi_{2^j}(u - 2^{-j}n) > = \sum_{-\infty}^{+\infty} \tilde{g}(2n - k) < f(u) , \phi_{2^j+1}(u - 2^{-j-1}k) > \quad (27)$$

Equation (27) shows that we can compute the detail signal $D_{2^j}f$ by convolving $A^d_{2^j+1}f$ with the filter $\tilde{G}$ and keeping every other sample of the output. The orthogonal wavelet representation of a discrete signal $A^d_1 f$ can therefore be computed by successively decomposing $A^d_{2^j+1}f$ into $A^d_{2^j}f$ and $D_{2^j}f$ for $-J \le j \le -1$. This algorithm is illustrated by the block diagram shown in Figure 5. In practice the signal $A^d_1 f$ has only a finite number of samples. One method of handling the border problems uses a symmetry with respect to the first and last samples.

Equation (18) of Theorem 3 can be used to show that the impulse response of the filter G is related to the impulse response of the filter H by

$$g(n) = (-1)^{1-n} h(1-n) \quad (28)$$

G is the mirror filter of H and is a high–pass filter. G and H are called

quadrature mirror filters. Equation (27) can be interpreted as a high–pass filtering of the discrete signal $A^d_{2^j+1}f$.

If the original signal has N samples, then the discrete signals $D_{2^j}f$ and $A^d_{2^j}f$ have 2^jN samples each. Thus the wavelet representation

$$(A^d_{2^{-J}}\, f\ ,\ (D_{2^j}f)_{-J \le j \le -1}\ )$$

has the same total number of samples as the original approximated signal $A^d_1 f$. This occurs because the representation is orthogonal. The energy of the samples of $D_{2^j}f$ gives a measure of the irregularity of the signal at the resolution 2^{j+1}. Whenever $A_{2^j}f(x)$ and $A_{2^j+1}f(x)$ are significantly different, the signal detail has a high amplitude.

2.6 SIGNAL RECONSTRUCTION FROM AN ORTHOGONAL WAVELET REPRESENTATION

Here we show that the original discrete signal can also be reconstructed with a pyramid transform. Since O_{2^j} is the orthogonal complement of V_{2^j} in V_{2^j+1},

$(\sqrt{2^{-j}}\ \phi_{2^j}(x - 2^{-j}n),\ \sqrt{2^{-j}}\ \Psi_{2^j}(x - 2^{-j}n))_{n \in Z}$ is an orthonormal basis of V_{2^j+1}. For any $n > 0$, the function $\phi_{2^j+1}(x - 2^{-j-1}n)$ can thus be decomposed in this basis

$$\begin{aligned}
&\phi_{2^j+1}(x - 2^{-j-1}n) \\
&= 2^{-j} \sum_{-\infty}^{+\infty} < \phi_{2^j}(u - 2^{-j}k)\ ,\ \phi_{2^j+1}(u - 2^{-j-1}n) > \\
&\qquad \times \phi_{2^j}(x - 2^{-j}k)) \\
&+ 2^{-j} \sum_{-\infty}^{+\infty} < \Psi_{2^j}(u - 2^{-j}k)\ ,\ \phi_{2^j+1}(u - 2^{-j-1}n) > \\
&\qquad \times \Psi_{2^j}(x - 2^{-j}k))
\end{aligned} \tag{29}$$

By computing the inner product of each side of equation (29) with the function f(x), we have

$$\begin{aligned} & < f(u) \ , \ \phi_{2j+1}(u - 2^{-j-1}n) > \\ = \ & 2^{-j} \sum_{-\infty}^{+\infty} < \phi_{2j}(u - 2^{-j}k) \ , \ \phi_{2j+1}(u - 2^{-j-1}n) > \\ & \times < f(u) \ , \ \phi_{2j}(u - 2^{-j}k) > \\ + \ & 2^{-j} \sum_{-\infty}^{+\infty} < \Psi_{2j}(u - 2^{-j}k) \ , \ \phi_{2j+1}(u - 2^{-j-1}n) > \\ & \times < f(u) \ , \ \Psi_{2j}(u - 2^{-j}k) > \end{aligned} \tag{30}$$

Inserting (13) and (24) in this expression and using the filters H and G respectively, defined by (14) and (26) yields

$$\begin{aligned} & < f(u) \ , \ \phi_{2j+1}(u - 2^{-j-1}n) > \\ & = 2 \sum_{-\infty}^{+\infty} h(n - 2k) < f(u) \ , \ \phi_{2j}(u - 2^{-j}k) > \\ & + 2 \sum_{-\infty}^{+\infty} g(n - 2k) < f(u) \ , \ \Psi_{2j}(u - 2^{-j}k) > \end{aligned} \tag{31}$$

This equation shows that $A^d_{2j+1}f$ can be reconstructed by putting zeros between each sample of $A^d_{2j}f$ and $D_{2j}f$ and convolving the resulting signals with the filters H and G respectively. The original discrete signal $A^d_1 f$ at the resolution 1 is reconstructed by repeating this procedure for $-J \leq j < 0$.

2.7 EXTENSION OF THE ORTHOGONAL WAVELET REPRESENTATION TO IMAGES

The Wavelet model can be easily generalized to any dimension $n > 0$ as shown by the work of Y.Meyer. Here, we are interested in the two–dimensional case of image processing applications. The signal is now a finite energy function $f(x,y) \in L^2(R^2)$. A multiresolution approximation of $L^2(R^2)$ is a sequence of subspaces $L^2(R^2)$ which satisfies a straightforward two–dimensional extension of the properties (2)–(8). Let $(V_{2j})_{j \in Z}$ be such a multiresolution approximation of

$L^2(R^2)$ is a sequence of subspaces $L^2(R^2)$ which satisfies a straightforward two–dimensional extension of the properties (2)–(8). Let $(V_{2^j})_{j \in Z}$ be such a multiresolution approximation of $L^2(R^2)$. The approximation of a signal f(x,y) at a resolution 2^j is equal to its orthogonal projection on the vector space V_{2^j}. Theorem 1 is still valid in two dimensions and it can be shown that there exists a unique scaling function $\phi(x,y)$ whose dilation and translation gives an orthonormal basis of each space V_{2^j}. Let $\phi_{2^j}(x,y) = 2^{2j}\phi(2^j x, 2^j y)$. The family of functions

$$(2^{-j}\, \phi_{2^j}(x - 2^{-j}n,\, y - 2^{-j}m))_{(n,m) \in Z^2}$$

forms an orthonormal basis of V_{2^j}. The factor 2^{-j} normalizes each function in the $L^2(R^2)$ norm. The function $\phi(x,y)$ is unique with respect to a particular multiresolution approximation of $L^2(R^2)$.

We consider the particular case of separable multiresolution approximations of $L^2(R^2)$. For such multiresolution approximations, each vector space V_{2^j} can be decomposed as a tensor product of two identical subspaces $(V^1_{2^j})$ of $L^2(R)$. The sequence of vector spaces $(V_{2^j})_{j \in Z}$ forms a multiresolution approximation of $L^2(R^2)$, if and only if $V^1_{2^j}$ is a multiresolution approximation of $L^2(R)$. One can thus easily show that the scaling function $\phi(x,y)$ can be written as

$$\phi(x,y) = \phi(x)\, \phi(y)$$

where $\phi(x)$ is the one–dimensional scaling function of the multiresolution approximation $(V^1_{2^j})_{j \in Z}$. With a separable multiresolution approximation, extra importance is given to the horizontal and vertical directions in the image. For many types of images, such as those from man–made environments, this emphasis is appropriate. The orthogonal basis of V_{2^j} is then given by

$$\begin{aligned}&(2^{-j}\, \phi_{2^j}(x - 2^{-j}n,\, y - 2^{-j}m))_{(n,m) \in Z^2} \\ &\quad = (2^{-j}\, \phi_{2^j}(x - 2^{-j}n)\, \phi_{2^j}(y - 2^{-j}m))_{(n,m) \in Z^2} \qquad (32)\end{aligned}$$

The approximation of a signal f(x,y) at a resolution 2^j is therefore characterized by a set of inner products

$$A^d_{2^j} f = $$

$$(< f(x,y) , \phi_{2^j}(x - 2^{-j}n)\ \phi_{2^j}(y - 2^{-j}m) >)_{(n,m) \in Z^2}$$

Assuming that the camera measures an approximation of the irradiance of a scene at the resolution 1, let $A^d_1 f$ be the resulting image and N be the number of pixels. One can easily show that for $j < 0$, a discrete approximation $A^d_{2^j} f$ has $2^j N$ pixels. Border problems are handled by supposing that the original image is symmetric with respect to the horizontal and vertical borders.

As in the one–dimensional case, the detail signal at the resolution 2^j is equal to the orthogonal projection of the signal on the orthogonal complement of V_{2^j} in V_{2^j+1}. Let O_{2^j} be this orthogonal complement. The following Theorem gives a simple extension of Theorem 3, and states that we can build an orthonormal basis of O_{2^j} by scaling and translating three wavelet functions, $\Psi^1(x,y)$, $\Psi^2(x,y)$ and $\Psi^3(x,y)$.

Theorem 4 : Let $(V_{2^j})_{j \in Z}$ be a separable multiresolution approximation of $L^2(R^2)$. Let $\phi(x,y) = \phi(x)\,\phi(y)$ be the associated two–dimensional scaling function. Let $\Psi(x)$ be the one–dimensional wavelet associated with the scaling function $\phi(x)$. Then, the three "wavelets"

$$\Psi^1(x,y) = \phi(x)\Psi(y) ,$$
$$\Psi^2(x,y) = \Psi(x)\phi(y) ,$$
$$\Psi^3(x,y) = \Psi(x)\Psi(y).$$

are such that

$$(2^{-j}\ \Psi^1_{2^j}(x - 2^{-j}n,\ y - 2^{-j}m),$$
$$2^{-j}\ \Psi^2_{2^j}(x - 2^{-j}n,\ y - 2^{-j}m),$$
$$2^{-j}\ \Psi^3_{2^j}(x - 2^{-j}n,\ y - 2^{-j}m))_{(n,m) \in Z^2} \qquad (33)$$

is an orthonormal basis of O_{2^j} and

$$(2^{-j}\ \Psi^1_{2^j}(x - 2^{-j}n,\ y - 2^{-j}m),$$
$$2^{-j}\ \Psi^2_{2^j}(x - 2^{-j}n,\ y - 2^{-j}m),$$
$$2^{-j}\ \Psi^3_{2^j}(x - 2^{-j}n,\ y - 2^{-j}m))_{(n,m)\ \in\ Z^3} \qquad (34)$$

is an orthonormal basis of $L^2(R^2)$.

The difference of information between $A^d_{2^{j+1}}f$ and $A^d_{2^j}f$ is equal to the orthonormal projection of f(x) on O_{2^j}, and is characterized by the inner products of f(x) with each vector of an orthonormal basis of O_{2^j}. Theorem 4 says that this difference of information is given by the three detail images

$$D^1_{2^j}f = (\ <\ f(x,y)\ ,\ \Psi^1_{2^j}(x - 2^{-j}n,\ y - 2^{-j}m)\ >\ )_{(n,m)\ \in\ Z^2} \qquad (35)$$

$$D^2_{2^j}f = (\ <\ f(x,y)\ ,\ \Psi^2_{2^j}(x - 2^{-j}n,\ y - 2^{-j}m)\ >\ )_{(n,m)\ \in\ Z^2} \qquad (36)$$

$$D^3_{2^j}f = (\ <\ f(x,y)\ ,\ \Psi^3_{2^j}(x - 2^{-j}n,\ y - 2^{-j}m)\ >\ )_{(n,m)\ \in\ Z^2} \qquad (37)$$

Just as for one–dimensional signals, one can show that in two dimensions the inner products which define $A^d_{2^j}f$,$D^1_{2^j}f$, $D^2_{2^j}f$ and $D^3_{2^j}f$ are equal to a uniform sampling of two–dimensional convolution products. Since the three wavelets $\Psi^1(x,y)$, $\Psi^2(x,y)$ and $\Psi^3(x,y)$ are given by separable products of the functions ϕ and Ψ, these convolutions can be written

$$A^d_{2^j}f =$$
$$((\ f(x,y)\ *\ \phi_{2^j}(-x)\phi_{2^j}(-y))(\ 2^{-j}n\ ,\ 2^{-j}m))_{(n,m)\ \in\ Z^2} \qquad (38)$$

$$D^1_{2^j}f =$$
$$((\ f(x,y)\ *\ \phi_{2^j}(-x)\Psi_{2^j}(-y))(\ 2^{-j}n\ ,\ 2^{-j}m))_{(n,m)\ \in\ Z^2} \qquad (39)$$

$$D^2_{2^j}f =$$
$$((\ f(x,y)\ *\ \Psi_{2^j}(-x)\phi_{2^j}(-y))(\ 2^{-j}n\ ,\ 2^{-j}m))_{(n,m)\ \in\ Z^2} \qquad (40)$$

$$D^3_{2^j}f =$$
$$((\ f(x,y)\ *\ \Psi_{2^j}(-x)\Psi_{2^j}(-y))(\ 2^{-j}n\ ,\ 2^{-j}m))_{(n,m)\ \in\ Z^2} \qquad (41)$$

The expressions (38) through (41) show that in two dimensions, $A^d_{2^j}f$ and the $D^k_{2^j}f$ are computed with separable filtering of the signal along the abscissa and ordinate.

The wavelet decomposition can thus be interpreted as a signal decomposition in a set of independent, ***spatially oriented*** channels. Let us suppose that $\phi(x)$ and $\Psi(x)$ are, respectively, a perfect low–pass and a perfect band pass filter. Figure 2.3 shows in the frequency domain how the image $A^d_{2^{j+1}}f$ is decomposed into $A^d_{2^j}f$, $D^1_{2^j}f$, $D^2_{2^j}f$ and $D^3_{2^j}f$. The image $A^d_{2^j}f$ corresponds to the lowest frequencies, $D^1_{2^j}f$ gives the vertical high frequencies (horizontal edges), $D^2_{2^j}f$ the horizontal high frequencies (vertical edges) and $D^3_{2^j}f$ the high frequencies in both directions (the corners). The arrangement of the $D^k_{2^j}f$ images is shown in Figure 10(b).

For any $J > 0$, an image A^d_1f is completely represented by the (3J+1) discrete images $(A^d_{2^{-J}}f$, $(D^1_{2^j}f)_{-J \le j \le -1}$, $(D^2_{2^j}f)_{-J \le j \le -1}$ and $(D^3_{2^j}f)_{-J \le j \le -1})$. This set of images is called an ***orthogonal wavelet representation*** in two dimensions. The image $A^d_{2^{-J}}f$ is the coarse approximation at the resolution 2^{-J} and the $D^k_{2^j}f$ images give the detail signals for different orientations and resolutions. If the original image has N pixels, each image $A^d_{2^j}f$, $D^1_{2^j}f$, $D^2_{2^j}f$, $D^3_{2^j}f$ has 2^jN pixels ($j < 0$). The total number of pixels in this new representation is equal to the number of pixels of the original image, so we do not increase the volume of data. Once again, this occurs due to the orthogonality of the representation. In a correlated multiresolution representation such as the Laplacian pyramid, the total number of pixels representing the signal is increased by a factor of 2 in one dimension and of 4/3 in two dimensions.

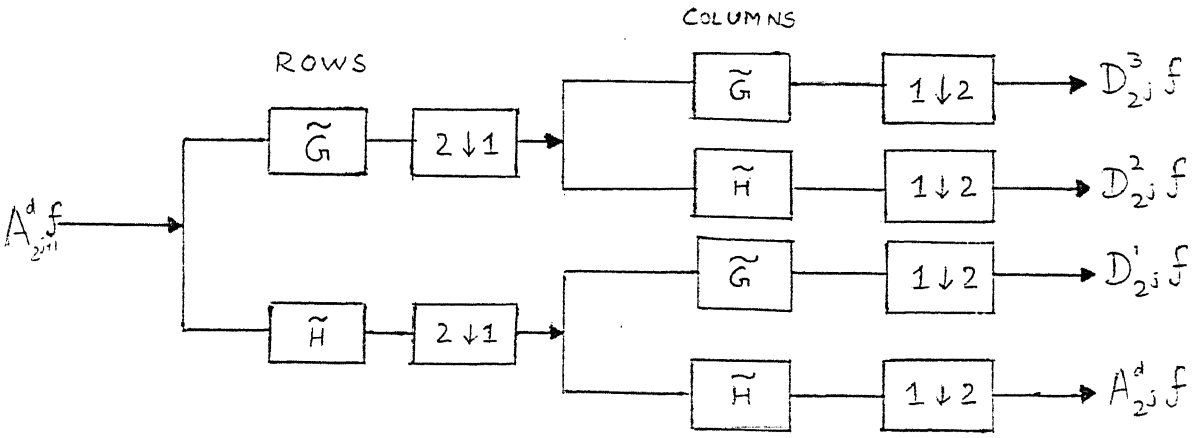
2.8 DECOMPOSITION AND RECONSTRUCTION ALGORITHMS IN TWO DIMENSIONS

In two dimensions, the wavelet representation can be computed with a pyramidal algorithm similar to the one–dimensional algorithm. The two dimensional wavelet transform can be seen as a one–dimensional wavelet transform along the x and y axes. It can be shown that a two–dimensional wavelet transform can be computed with a separable extension of the one–dimensional decomposition algorithm. At each step we decompose $A^d_{2^{j+1}}f$ into $A^d_{2^j}f$, $D^1_{2^j}f$,$D^2_{2^j}f$,and $D^3_{2^j}f$. This algorithm is illustrated by the block diagram in Figure 2.1. We first convolve the rows of $A^d_{2^{j+1}}f$ with a one–dimensional filter, retain every other row, convolve the columns of the resulting signals with another one–dimensional filter and retain every other column. The filters used in this decomposition are the quadrature mirror filters $\tilde{H}$ and $\tilde{G}$.

The structure of application of the filters is given in the Figure 2.1. We compute the wavelet transform of an image A^d_1f by repeating this process for $-1 \geq j \geq -J$. This corresponds to a separable conjugate mirror filter decomposition. The wavelet coefficients have a high amplitude around the images edges and in the textured areas within a given spatial orientation.

The one–dimensional reconstruction algorithm can also be extended to two dimensions. At each step, the image $A^d_{2^{j+1}}f$ is reconstructed from $A^d_{2^j}f$,$D^1_{2^j}f$,$D^2_{2^j}f$,and $D^3_{2^j}f$. This algorithm is illustrated in Figure 2.2. Between each column of the images $A^d_{2^j}f$,$D^1_{2^j}f$,$D^2_{2^j}f$,and $D^3_{2^j}f$, we add a column of zeros, convolve the rows with one–dimensional filter, add a row of zeros between each row of the resulting image, and convolve the columns with another one–dimensional filter. The filters used in reconstruction are the quadrature mirror filters H and G. The

image $A_1^d f$ is reconstructed from its wavelet transform by repeating this process for $-J \leq j \leq -1$. If we use floating–point precision for the discrete signals in the wavelet representation, the reconstruction is of excellent quality Figure 2.2 shows the reconstruction of the original image from its wavelet representation.

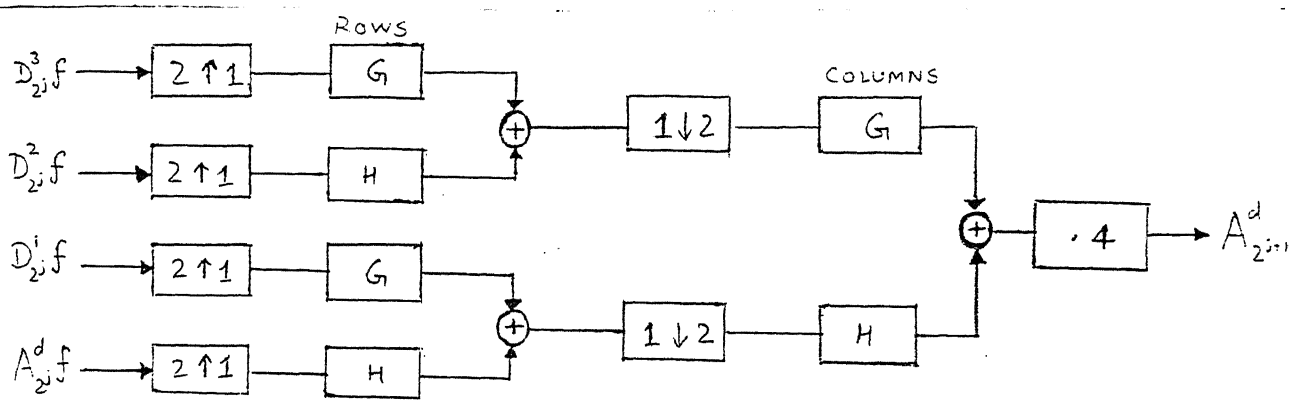


[x] : Convolve (rows or columns with filter x).

[2↓1] : Keep one column out of two

[1↓2] : Keep one row out of two .

Fig. 2.1. Decomposition of an image $A^d_{2^{j+1}} f$ into the coarse signal $A^d_{2^j} f$, $D^1_{2^j} f$, $D^2_{2^j} f$, and $D^3_{2^j} f$. This algorithm is based on one–dimensional convolutions of the rows and columns of $A^d_{2^{j+1}} f$ with the one–dimensional quadrature mirror filters $\tilde{H}$ and $\tilde{G}$.



X : Convolve (rows or columns) with filter X

2↓1 : Put one column of zeros between each column

1↓2 : Put one row of zeros between each row

·4 : Multiply by 4.

Fig. 2.2. Reconstruction of an image $A^d_{2^{j+1}}f$ from $A^d_{2^j}f$, $D^1_{2^j}f$, $D^2_{2^j}f$, and $D^3_{2^j}f$. The rows and columns of these images are convolved with the one–dimensional quadrature mirror filters H and G.

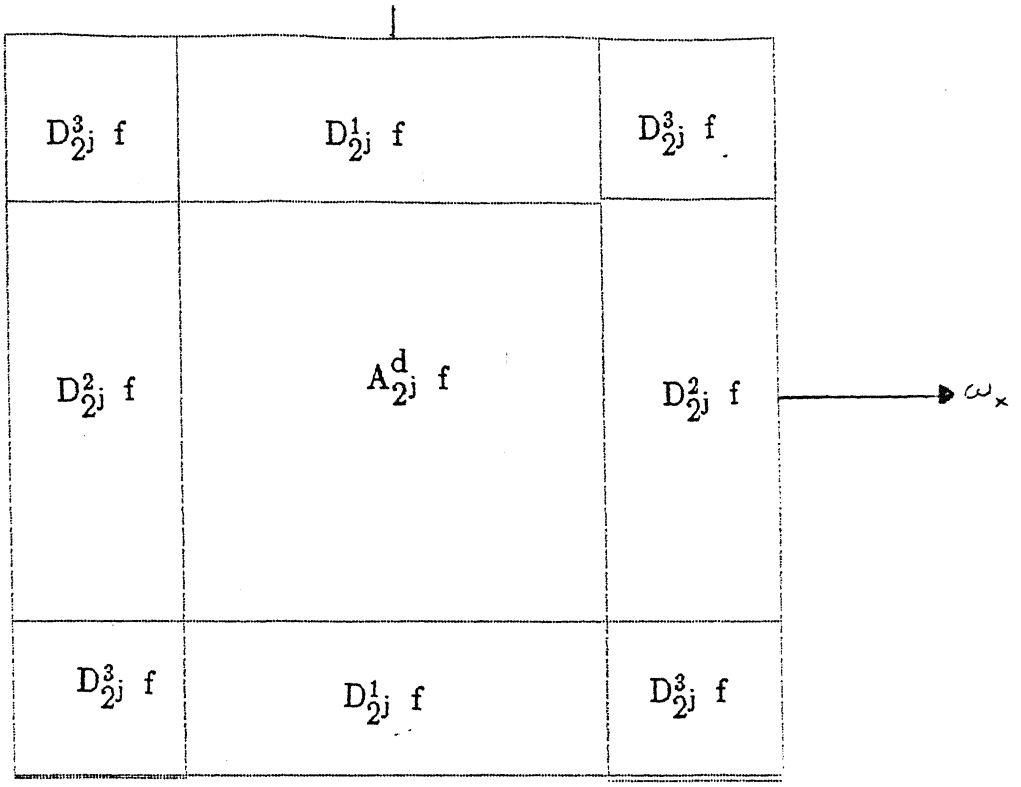


Fig. 2.3 (a). Decomposition of the frequency support of the image $A^d_{2^j+1} f$ into $A^d_{2^j} f$ and the detail images $D^k_{2^j} f$.

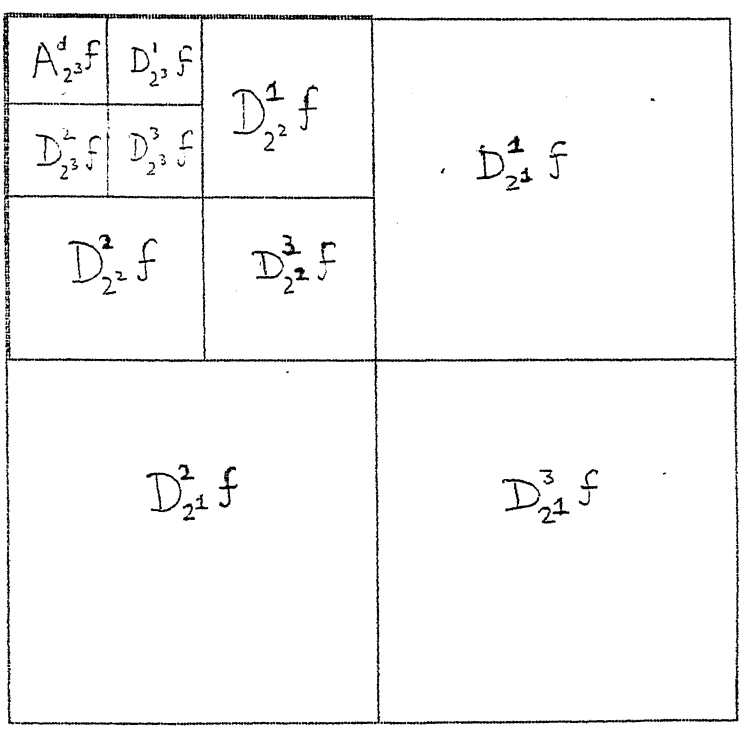


Fig. 2.3(b). Disposition of the $D^k_{2^j} f$ and $A^d_{2^j} f$ images of the wavelet representation.

CHAPTER 3

TRANSPUTER AND OCCAM

3.1. TRANSPUTER

3.1.1. Introduction:

The word *Transputer* may be interpreted as a contraction of the word *Tranceiver* and *computer*. The interpretation suggests that the Transputer consist of a communication system and a computational element.

"Transputer" can be used as a basic element in multiprocessor systems. Th chip, being very versatile, has received considerable attention and popularity amon concurrent system designers recently.

There are a number of interesting features of the Transputers whic distinguish the Transputer from any other processor. Of all, the integration of th design and the advantage of having many features built into a single chip are th important aspects, which result in the enhancement of processor speed, low chi count, and simplicity of the system design.

A Transputer can be used in a single processor system or in networks t build high performance concurrent systems. A typical member of transputer produc family is a single chip containing processor, memory and point–to–poin communication links. A network of Transputers can be easily constructed usin these links. As a microcomputer, the Transputer is unusual in its ability t communicate with other Transputers. A variety of different configurations can b built by hard–wiring Transputers together, with no separate switching an

forwarding network, limited only by the number of links provided on each Transputer. The current Transputer systems have four to six links. Four links are enough to allow enormous range of useful configurations.

Scalability is a major advantage of transputer–based systems: it is much easier to enhance a system by adding further processors to it than would be the case with other microprocessors. ***Compatibility*** – the ease of replacing one transputer model with another without major design changes in a system is also valuable. This extends to mixing models of transputer within one system.

3.1.2. Architectural details :

Transputers are often described as RISC processors. The computational instructions follow RISC principles closely, and they attain the benefits claimed for RISC architecture. However, they also have a small number of important non–RISC instructions concerned with scheduling and message passing.

The Transputer is unusual in its ability to execute many software processes at the same time. A program can be run on a single Transputer, in which case the concurrency of the processes will be simulated by hardware with no software intervention. Provided the communication between the subprocesses is not too complicated, the same program can also be distributed over several processors, in which case the component processes will be run in real concurrency. Just as in a single processor, interprocess message passing and the necessary synchronization (between directly connected Transputers) are achieved in hardware, and no operating system is needed.

The inbuilt features of the Transputer make up a long list. As well as the instruction processor and small amount of on chip memory, it has a memory controller, DMA control for four independent fast links, a microcoded multitasking

kernel. and an elapsed–time clock (Fig. 3.1). The best known Transputers are the T212, T414 and T800. The T212 is a 16–bit processor, the other two are 32–bit processors. The T800 has a full IEEE floating–point processor on chip.

3.1.2.1. Instruction processor :

The processor portion of a Transputer is a traditional microprocessor. Fig 3.2 shows the 32–bit version of the Transputer. The processor normally obtains its instructions and data from the internal 4K RAM .Data and instructions can also be obtained from the links. The processor provides 32–bit addressing. memory is addressed byte–wise and stored in 4–byte units. Software sees no difference between on–chip and external memory except in speed.

The design of Transputer processor exploits the availability of fast on–chip memory by having only a small number of registers; the CPU contains six registers which are used in the execution of a sequential process. The small number of registers, together with the simplicity of the instruction set enables the processor to have relatively simple (and fast) data–paths and control logic.

The instructions refer to the stack implicitly. The use of a stack removes the need for instructions to respecify the location of their operands. Statistics gathered from a large number of programs show that three registers provide an effective balance between code compactness and implementation complexity.

3.1.2.1. Instruction Set:

Transputer instruction set is designed for simple and efficient compilation. All the instructions have the same format and chosen to give a compact representation of the operations most frequently occurring in programs. The instruction size of the Transputer is only 8 bits for most instructions. There are prefix instructions which allow the operand to be extended to any length.

Measurements show that about 70% of the executed instructions are encoded in a single byte. There is an extra word of prefetch buffer, so the processor rarely has to wait for an instruction fetch before proceeding.

3.1.2.3. Memory controller :

The memory controller can drive external dynamic RAM with no additional circuitry. Together with the controller, the processor can address a linear address space of 4–Gbytes. The 32–bit wide memory interface uses multiplexed data and address lines and provides a data rate of up to 4 bytes every 100 nanoseconds (40 Mbytes/sec) for a 30 MHz device. The configurable memory controller provides all timing, control and DRAM refresh signals for a wide variety of mixed memory systems.

3.1.2.4. Process scheduler :

The processor provides efficient support for concurrency and communication. It has a microcoded scheduler which enables any number of concurrent processes to be executed together, sharing the processor time. This removes the need for a software kernel. The processor does not need to support the dynamic allocation of the storage as the compiler is able to perform the allocation of space to the concurrent processes.

A process starts, performs a number of actions, and then terminates. Typically a process is a sequence of instructions. A transputer can run several processes in parallel (concurrently). Processes may be assigned either high or low priority. At any time, a concurrent process may be

active — being executed

— on a list waiting to be executed

inactive — ready to input

— ready to output

— waiting until a specified time

The scheduler operates in such a way that inactive processes do not consume any processor time. The active processes waiting to be executed are held on a list. The IMS T800 supports two levels of priority. Priority 1 (low priority) processes are executed whenever there are no active priority 0 (high priority) processes.

High priority processes are expected to execute for a short time. If one or more high priority processes are able to proceed, then one is selected and run until it has to wait for communication, a timer input, or until it completes processing.

Each process runs until it has completed its action, but is descheduled whilst waiting for communication from another process or Transputer, or for a time delay to complete. In order for several processes to operate in parallel, a low priority process is only permitted to run for a maximum of two time slices before it is forcibly descheduled at the next descheduling point.

A process can only be descheduled on certain instructions, known as descheduling points. As a result, an expression evaluation can be guaranteed to execute without the process being time sliced part way through. Actual process switch times are less than 1 μs, as little state needs to be saved and it is not necessary to save the evaluation stack on rescheduling.

3.1.2.5. Communications :

Communication between the processes is achieved by means of channels. The communication is point–to–point, synchronized and unbuffered. As a result, a channel needs no process queue, no message queue and no message buffer.

A channel between two processes executing on the same Transputer is implemented by a single word in memory; a channel between processes executing on different Transputers is implemented by point–to–point links. The processor provides a number of operations to support message passing, the most important being

input message ***output message***

The input message and output message instructions use the address of the channel to determine whether the channel is internal or external. This means that the same instruction sequence can be used for both for hard and soft channels, allowing a process to be written and compiled without the knowledge of where its channels are connected.

The communication takes place when both the inputting and outputting processes are ready. Consequently, the process which first becomes ready must wait until the second one is also ready.

A process performs an input or output by loading the evaluation stack with a pointer to a message, the address of a channel, and a count of the number of bytes to be transferred, and then executing an input message or an output message instruction.

3.1.2.5.1. Internal channel communication :

At any time, an internal channel (a single word in memory) either holds the identity of a process, or holds the special value *empty*. The channel is initialized to *empty* before it is used.

when a message is passed using the channel, the identity of the first process to become ready is stored in the channel, and the process starts to execute the next process from the scheduling list. When the second process to use the channel

becomes ready, the message is copied, the waiting process is added to the scheduling list, and the channel reset to its initial state. It does not matter whether the inputting or the outputting process becomes ready first.

3.1.2.5.2. External channel communication :

When a message is passed via an external channel the processor delegates to an autonomous link interface the job of transferring the message and deschedules the process. When the message has been transferred, the link interface causes the processor to reschedule the waiting process. This allows the processor to continue the execution of the other processes whilst the external message transfer is taking place.

3.1.2.6. Communication links :

A link between two transputers is implemented by connecting a link interface on one Transputer to a link interface on the other Transputer by two one–directional signal wires, along which data is transmitted serially. The two wires provide two channels, one in each direction. This requires a simple protocol to multiplex data and control information. Messages are transmitted as a sequence of bytes, each of which are to be acknowledged before the next is transmitted. A byte of data is transmitted as a start bit followed by a one bit followed by eight bits of data followed by a stop bit. An acknowledgement is transmitted as a start bit followed by a stop bit. An acknowledgement indicates that a process was able to receive the data byte and that it is able to buffer another byte.

The protocol permits an acknowledgement to be generated as soon as the receiver has identified a data packet. In this way, the acknowledgement can be received by the transmitter before all of the data packets have been transmitted

and the transmitter can transmit the next data packet immediately. The IMS T414 transputer does not implement this overlapping and achieves a data rate of 0.8 Mbytes per second using a link to transfer in one direction. However, by implementing sufficient overlapping and including sufficient buffering in the link hardware, the IMS T800 more than doubles this data rate to 0.8 Mbytes per second in one direction. and achieves 2.4 Mbytes per second when the link carries data in both directions.

3.1.2.7. The Timer :

The Transputers have two timer clocks which 'tick' periodically. The timer provide accurate process timing, allowing processes to be descheduled until a specific time.

In the IMS T800 Transputer, one timer is accessible to only high priority processes and is incremented every micro second, cycling completely in approximately 4295 milliseconds. The other is accessible only to a low priority process and is incremented every 64 microseconds, giving exactly 15625 ticks in one second. It has a full period of approximately 76 hours.

3.1.2.8. The Floating point processor :

The IMS T800 has a full IEEE floating point processor on chip. The FPU operates concurrently with CPU. This means that it is possible to do address calculation in the CPU whilst the FPU performs the floating point calculation. This can lead to significant performance improvements in real applications which access arrays heavily. Performance depends on many things, including clock and memory speeds — for the 20 MHz T800 these figures are of the order of 10 [illegible]0 MIPS and 1 MFLOPS (these are not upper bounds).

As can be noticed from above, all the links can be active at the same time as well as the processor. Thus the Transputer can support nine truly concurrent activities. (one link can transfer data in both directions, of course, memory accesses have to be interleaved). On a T800, the floating point processor operate in parallel with the instruction processor, which gives a tenth level of concurrency at the hardware level (but both the processors are controlled by a single instruction stream).

3.2. THE OCCAM

Occam's Razor – ***Entities are not to be multiplied beyond necessity*** was the philosophy of the original implementation of Occam.

Occam is the native language for Transputers. The design of Occam has been heavily influenced by the work on ***Communicating Sequential Processes*** (CSP), which gives a mathematical frame work for specifying the behaviour of parallel processes. Occam is based on the CSP model of computation, but with features chosen to ensure efficiency of implementation. In this model, an application is decomposed into a collection of communicating processes, and the processes communicate by passing messages.

Occam model is based on the idea of ***process***. The software building block is a ***process***. A system is designed in terms of an interconnected set of processes. Each *process* cab be regarded as an independent unit of design. Its internal design is hidden and is completely specified by the message it sends and receives. Internally, each process can be designed as a set of communicating processes. The system design is therefore hierarchically structured. Occam processes do not share any variables, nor semaphores. Occam does not require (nor support) shared

memory. Messages pass from exactly one process to the other. There are no multiple senders or receivers, no broadcasting, and no uncertainty about where a message came from or where it is going. Messages are unbuffered, so sending and receiving a message involves momentary synchronization between the two participating processes. Messages are sent through static channels, as if through a circuit switched (rather than packet switched) network.

To gain the most benefit from the Transputer architecture, the whole system can be programmed in Occam. This provides all the advantages of a high level language, the maximum program efficiency and the ability to use the special features of the Transputer. The Occam model of concurrency is applicable equally to processes running on separate processors and to processes running within a single processor. Since the processor can be controlled only by one instruction stream, it is evident that ***the processes in one processor cannot be truly concurrent***. However, the processes can be multi programmed, just as on a mainframe, so that the effect of concurrency is reproduced (apart from speed). This 'simulated concurrency' as distinct from 'real concurrency' is known as '***gratutious concurrency***'. Within a Transputer, this multiprogramming is handled by hardware with no need for any operating system.

Occam provides a framework for designing concurrent systems using Transputers just in the same way that boolean algebra provides a framework for designing electronic systems from logic gates. The system designer's task is eased because of the architectural relationship between Occam and Transputer. A program running in a Transputer is formally equivalent to an Occam process, and so a network of Transputers can be described directly as an Occam program.

Occam, when compiled for execution on a Transputer, is ideal for embedded multiprocessor systems. Where it is required to exploit concurrency, but still to

activities. This is a useful abstraction. One would like to be able to express an algorithm in the form of a program which is independent of hardware, so that it could be subsequently be performed using many different networks of processors. Each implementation would need a specification of how many processors were needed, how they were to be connected, and which processes were to be installed on which processors, but the specification ought not to need any change in the program. The specification is called ***placement***.

In practice, Transputer programs are not completely independent of their ***placement***. Unless it is carefully designed, a program will only run on one particular network of Transputers, or on a small number of similar networks. To run on other networks, the program itself will have to be changed. Sometimes the changes will be minimal, but other cases may need extensive modifications. Programmers therefore have to make a conscious effort to write programs which can easily be run on different configurations.

3.3.3. Non—determinacy :

Sequential programmers are used to the idea of a ***bug***. There are solid bugs and intermittent bugs. Intermittent bugs are data dependent; a program run on the same inputs will work every time or it fails every time. Concurrent programming has a third type of bug: the bug that depends on the relative timing of concurrent processes. These are very often not repeatable, even if the program is rerun on the same data.

The problem arises because all the processors are allowed to run at their own speed. *There is no attempt to constrain the processors into a lock step.* Thus the order of events can change from one test run to another.It is obviously important at the design phase not to assume an exact ordering of the events. Occam is as precise as any other language about what individual processes do, but

it cannot specify the relative timing of concurrent processes (otherwise the whole point of concurrency would be lost).

It is the programmer's responsibility to ensure that when a program terminates it has completed the required function, regardless of the order in which things happened between starting and termination. Since Occam does not guarantee determinacy, it has been designed to express and handle non–determinacy very simply, flexibly, and elegantly.

3.3.4. Deadlock :

A classic problem in concurrent systems is ***deadlock***. This is an affliction whereby one part of a program is waiting for another to do something; the other is waiting for the first to do something else; and since both are waiting, neither can do what other expects. There may, of course, be a set of processes involved, rather than a pair.

All Transputer deadlocks are essentially the same in that they involve a closed chain of processes, each trying to communicate with another, but with no pair of them willing to participating in any one communication. There is an enormous variety of ways which may lead to deadlock, and that makes it hard to avoid. It is also difficult to analyze and hard to cure after it is found to occur in a program. Formal methods used are writing only simple code, supported by intuition and back–of–envelope sketches.

3.3.5. Concurrent algorithm design considerations :

Most of the concurrent algorithms are only loosely related to their sequential equivalents. The conversion of a sequential algorithm into concurrent algorithm, often called '*parallelization*' is too ill defined and difficult to be automated and is often attempted by hand. The following points should be given due consideration

while developing the concurrent algorithms.

3.3.5.1. Granularity :

The term '*granularity*', which is jargon much used among concurrent programmers, refers to the size of the task that are distributed as concurrent processes. In a matrix multiplication, all the elements of the result can, in principle, be evaluated concurrently, because none of the calculations depends on the result of any other. That would be ***fine grain concurrency***. A more ***coarse grain*** division of labour could evaluate one quarter of the result, working sequentially on one processor, while evaluating the other three quarters on three other processors.

Intuitively, one would expect fine–grained concurrency to deliver results faster but to use more processors. With the Transputers, a ***finer–grain*** division of work requires more information to be passed between processors. Reducing the grain size below some optimum value for a particular problem can incur message passing costs which grossly outweigh the expected benefits.

Granularity is a particular problem in the conversion of existing sequential code into concurrent methods. It is relatively easy to recognize fine–grain potential parallelism, e.g.several assignments whose order is immaterial, or a simple loop. It is much harder to identify the larger units of a program which might safely be run concurrently.

3.3.5.2. Performance measure and efficiency :

One measure of the quality of a concurrent system is its efficiency in using the processors. If a single processor solution takes n seconds, it is desirable to achieve a solution with m processors in n/m seconds. Complete efficiency is never

attainable, but the user tries to get near to it. Efficiency must fall off as more processors are added.

Occasionally it does happen that an n–processor mechanism solves a problem in less than m/n seconds, which require m seconds on a single processor. This appears to show more than 100% efficiency. There are three possible explanations. One is that the speed is data dependent: if several sets of data are tried, all of which require m seconds on one processor, then some may need much less than m/n seconds on one processors, but others will need more than m/n seconds, so that the average performance is enhanced by a factor less than n. Characteristically, these problems have extremely variable solutions even on a single processor; i.e., the time needed to process one set of data is not obviously related to the time needed by other sets of data of apparently similar difficulty.

Another possible explanation is that, in recasting an algorithm for concurrent running, a better method than the original program may have been developed. For example, the sequential program might be doing more page swapping than the concurrent algorithm which result in the ***superlinear efficiency***.

It seems, then, that it sometimes possible for practical Transputer users to realize gains in performance larger than the number of processors used, but there is always some excuse for denying that this shows more than 100% efficiency. This suggests that a more sophisticated measure of efficiency is needed. In reality, the user should never expect to benefit from the super efficiency, for that occurs only rarely.

3.3.5.3. Balancing communication and processing :

Some times, the concurrent algorithm may show very poor efficiency. The reasons could be

- The computation has not been divided equally — one of the processor still has to do far more than 1/n th of the work.
- The processes are independent; they have to share or communicate information via links, and some of them spend a lot of time waiting for others, during which time they cannot continue with the computation.
- Even if time is not wasted in waiting for messages, there may lot of time spent in passing the messages.

These three possibilities are quite distinct. The first one has to be solved by '*load balancing*' — attempting to equalize the computation load on each processor. The second can be described as '*spurious concurrency*'. It can occur by accident, and it is not easy to predict, analyze or detect. Even when it is known that it is happening, it is not easily cured. The third reason for inefficiency is a difficult optimization problem between computation and communication times.

3.3.5.4. Software development :

Transputer software is mostly developed under the '*Transputer development environment*' (TDS) supplied by INMOS.

TDS and related systems provide an integrated environment for editing, compiling and running the programs. They are centered on the '*folding editor*' which embodies an elegant and general way of representing and handling large amount of Occam text within a small screen. They lack some of the support tools that are common in other systems (e.g.,in UNIX, diff, grep, multitasking, batch files, aliases, email, .log in files.)

The TDS is so much an '*integrated environment*' that its files are not easily handled in other systems, not even in systems such as MS–DOS which is acting

as the host or file server for TDS. This makes it hard for the utility software on the host system to provide the facilities that are missing from the TDS.

Programmers would very much like to have further help in the peculiar difficulties of concurrent program development. The strongest demand at the moment, and the one which seems nearest to fulfillment, is for software tools for profiling, monitoring, and debugging.

3.4 SETUP OF A TRANSPUTER NETWORK SYSTEM

The major components of a processor network are:

Host computer(mostly a PC)

Interface unit

processing element array(Transducers)

Interconnection networks

Host computer:

The host computer is intended to provide system monitoring, data storage and management. It generates global control codes and object codes of processor elements.It can be a microcomputer,work station, or a main frame. We have used PC–AT for this purpose. Processor elements can be accessed by a procedure call on the host, or through an interactive programmable command interpreter.

Interface unit:

Interface unit is an interface between the host and PE array. Interface unit, connected to the host via host or host bus, or DMA has the function of down loading,up loading, buffering array data and handling interrupts. It supports high bandwidth communication (accompanying high–speed processing)

between the array and the host. For balancing between low bandwidth of the system I/O and high bandwidth of the processor array, sufficient buffering is provided.

Processing element array :

A PE array consists of a number of processing elements with local memory. PE's effectively utilize its data storage thereby saving communication time.

Interconnection network :

Interconnection within PE array are provided by large switching networks which provide flexibility of interconnections and high speed communication between the PE's.

3.5 HARDWARE SETUP OF THE TRANSPUTER NETWORK AT IP–LAB, IITK.

A nine–node transputer network has been set up in the image processing lab at IIT Kanpur. The setup contains two IMS B003 evaluation boards, each having four transputers and an IMS B004 transputer system development(TDS) board having a transputer. Of these nine transputers, five are T800's and four are T414's. The host is an PC–AT (80386) One of the transputers functions as the interface unit, known as root processor, with 4 Mbytes of on board RAM. The root is connected to the host via a link adapter (IMS C012). Most of the time root transputer will be used for executing the I/O routines required for the host file server.PEs are individual transputers each with 256 Kbytes of RAM on board. Transputer link are connected by a programmable switch known as Link switch (IMS C004).

Link adapter :

It connects the host and the root transputer. It provides for full duplex transputer link communication by converting bi–directional serial link data into parallel data stream.

Host :

The host serves as a file server as memory management and file system are not supported by transputers. This facility of host is accessed by a program called server. The server reads a DOS file to determine the network configuration, the programs to be loaded and the boot order. The host loads the network via the root transputer by sending loading information to it, which in turn boots and loads the transputers connected to it which again passes the and loading information forward and so on until the whole network has been booted and loaded.

Booting of the transputer network :

A communication protocol exists between the host and the transputer network to direct the code to the desired place in each transputer. The bootstrap code for each transputer is sent first. After all transputers are booted, the code of each of the procedures allocated to processors is exported to the network preceded by necessary routing and loading information. Following this, the code which calls the procedures is sent to each processor.

The Linkswitch :

Transputer network is interconnected using Linkswitch (IMS C004).It is a transparent programmable linkswitch designed to provide a full crossbar switch between 32 link inputs and 32 link outputs. It uses the capabilities of VLSI to offer simple, easy to use and cheap interconnections for computer systems. It introduces on the average of 1.75 bit time delay on the signal. Linkswitches can

be cascaded to any depth without loss of signal integrity and can be used to construct configurable networks of arbitrary size. The switch is programmed via a separate link called the configuration link. In the setup, LINK0 of the root transputer is connected to the host (via link adapter) and LINK1 is used as the configuration link for the link switch. So, only LINK2 and LINK3 on the root transputer are free. Also,since the link switch supports only 32 Link connections, two links of the last (9th) transputer cannot be used at present (only LINK0 and LINK1 can be used on the last transputer).

3.6 CODE DEVELOPMENT

Occam is used for the program development under the TDS environment. The same program can be developed on a single node or on a multi node transputer network. On single node program would be developed as a parallel program of n processes. It would be using soft channels for communication between the parallel processes. On multi node network some changes have to be made:

1. The individual processes are to be define as procedures whose parameters should be only the hard channels and should be compiled separately.

2. A mapping of the Occam channels to the transputer hard links should be given. This gives the network configuration information.

3. The separately compiled procedures are 'PLACED' on individual transputers and the program is then compiled to generate a network code file.

For running the program, network linkswitch should be configured. This can be done by a software program which will transmit appropriate code to the configuration input link of the linkswitch.

The single/multi node program can be tested from the TDS environment. After successful run, it can be made a stand alone program bootable by the external host by using the alien file server library routines available in the TDS environment.

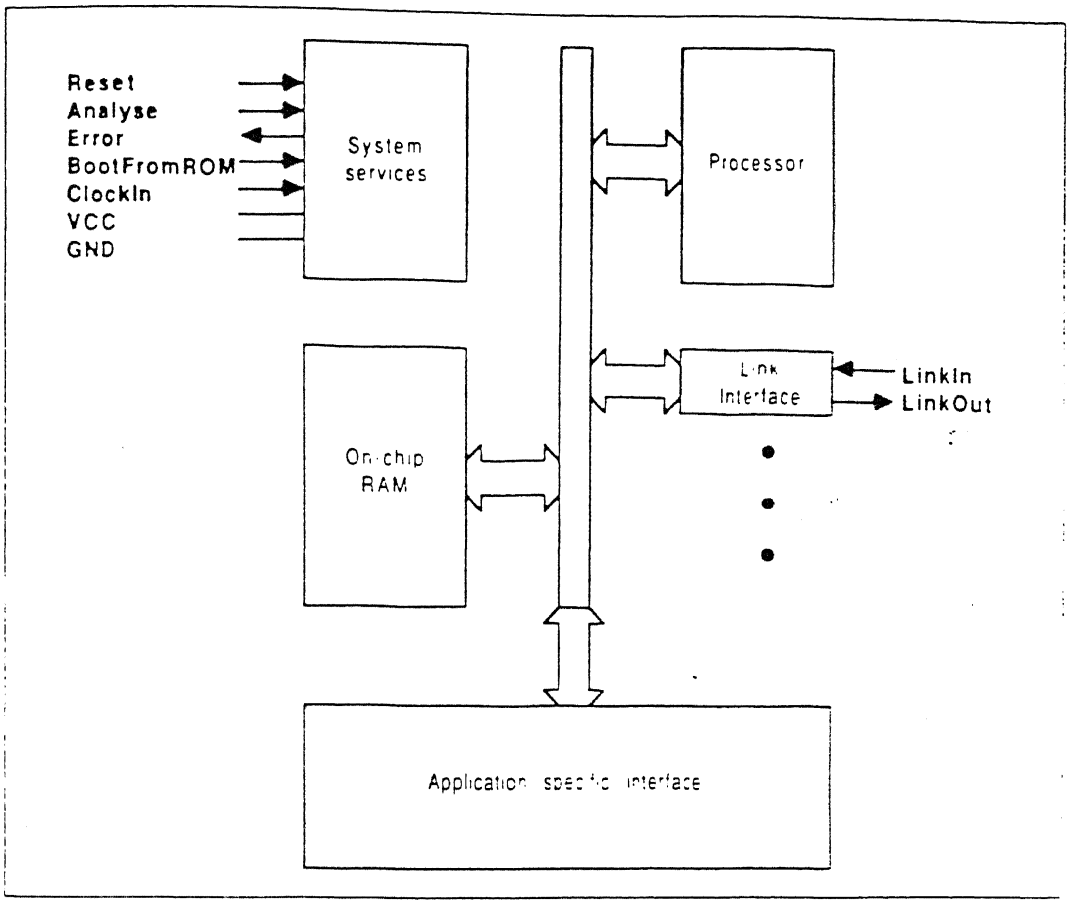


Fig.3.1: Transputer Architecture.

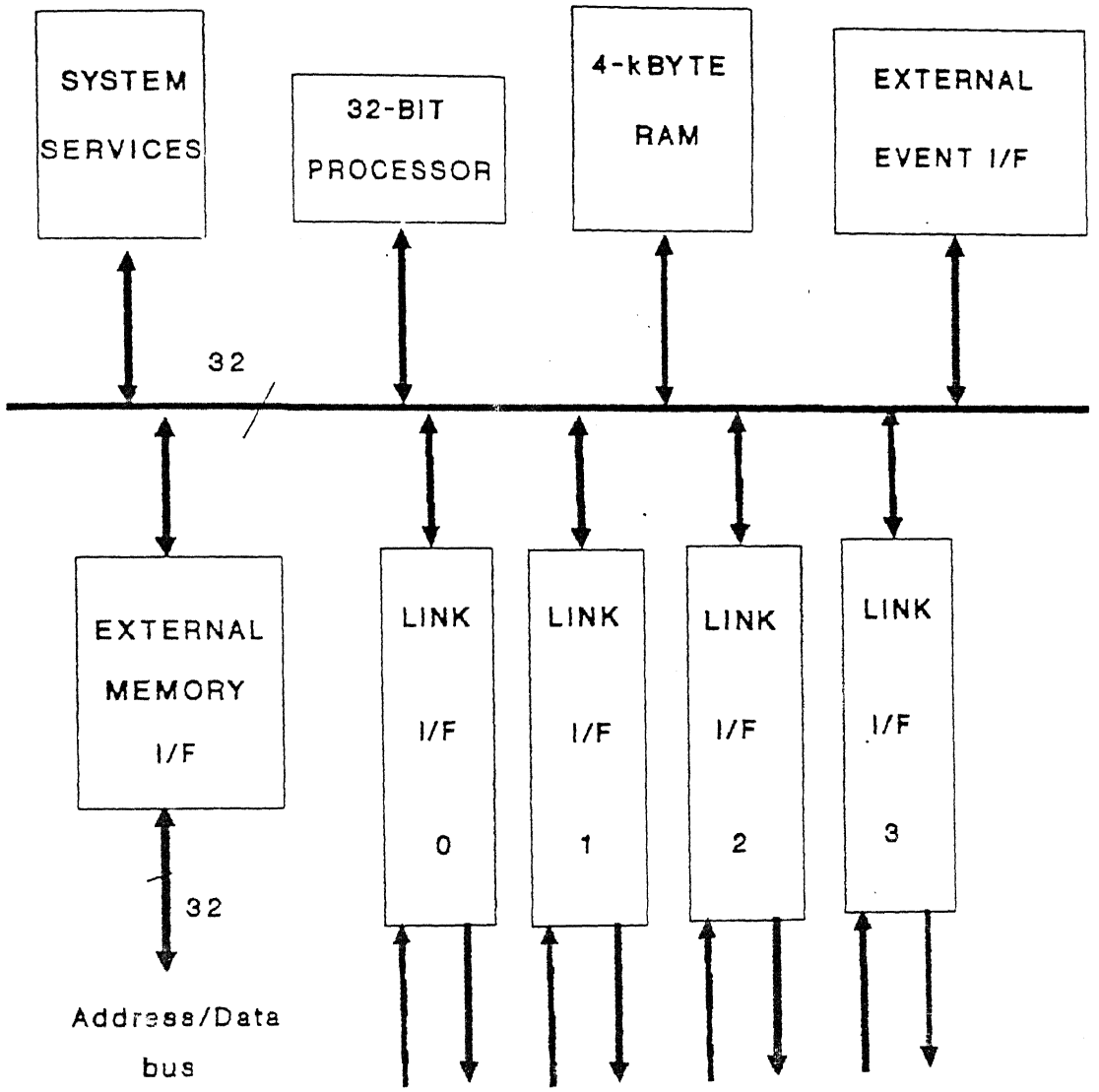


Fig. 3.2 INMOS T800 Transputer

CHAPTER 4

IMPLEMENTATION OF MULTIRESOLUTION ALGORITHMS ON A TRANSPUTER NETWORK

4.1 INTRODUCTION

The decomposition algorithm which has been developed for computing the multiresolution Wavelet Transform of images has already been discussed in Chapter 2. Similarly the reconstruction algorithm for getting the image back from the detail and coarse signals has also been studied in the same Chapter 2. In any image processing application, the time taken by the procedure is of crucial importance , especially so in real time applications. The concept of parallel processing comes in handy here. Our equipment consists of a root processor and 8 other processors. The root processor and 4 other processors are of the T800 type while the remaining 4 are of the T414 type. In this chapter, we shall discuss as to how these algorithms have been implemented using these Transputers.

4.2 THE ALGORITHMS

It would be useful to briefly recapitulate the decomposition and reconstruction algorithms. They are as follows:

A two–dimensional wavelet transform can be computed with a separable extension of the one dimensional decomposition algorithm. At each step we decompose $A^{d}_{2^{j+1}}f$ into $A^{a}_{2^{j}}f$, $D^{1}_{2^{j}}f$,$D^{2}_{2^{j}}f$,and $D^{3}_{2^{j}}f$. This algorithm is illustrated by

the block diagram in Figure 2.1. We first convolve the rows of $A^d_{2j+1}f$ with a one–dimensional filter, retain every other row, convolve the columns of the resulting signals with another one–dimensional filter and retain every other column. The filters used in this decomposition are the quadrature mirror filters $\tilde{H}$ and $\tilde{G}$.

The structure of application of the filters is given in the Fig. 2.1. We compute the wavelet transform of an image A^d_1f by repeating this process for $-1 \geq j \geq -J$. This corresponds to a separable conjugate mirror filter decomposition. The wavelet coefficients have a high amplitude around the images edges and in the textured areas within a given spatial orientation.

The one–dimensional reconstruction algorithm can also be extended to two dimensions. At each step, the image $A^d_{2j+1}f$ is reconstructed from $A^d_{2j}f$,$D^1_{2j}f$,$D^2_{2j}f$,and $D^3_{2j}f$. This algorithm is illustrated in Figure. 2.2. Between each column of the images $A^d_{2j}f$,$D^1_{2j}f$,$D^2_{2j}f$,and $D^3_{2j}f$, we add a column of zeros, convolve the rows with one–dimensional filter, add a row of zeros between each row of the resulting image, and convolve the columns with another one–dimensional filter. The filters used in reconstruction are the quadrature mirror filters H and G. The image A^d_1f is reconstructed from its wavelet transform by repeating this process for $-J \leq j \leq -1$. If we use floating–point precision for the discrete signals in the wavelet representation, the reconstruction is of excellent quality. Figure 2.2 shows the reconstruction of the original image from its wavelet representation.

4.3 TRANSPUTER NETWORK CONFIGURATION

Different configurations are possible when we use eight Transputers. The three possible configurations are shown in Fig. 4.1. However, in order to implement the above decomposition and reconstruction algorithms, we have used

the configuration depicted in Fig. 4.1(c). The configuration being considered is a linear array of the processors with data flow along a linear path. We have utilized the concept of data–partitioning or in parallel processing parlance, "*coarse grain concurrency*".

4.4 IMPLEMENTATION PROCEDURE

4.4.1 Image :

The image that we have taken up for consideration is the standard Baboon image. This particular image has been selected because the contrast between the various parts of the image is considerable. The size of the image that has been taken up for decomposition is 128×128 or smaller. This was so, because the system could not support the processing of larger images. In the description of the algorithm implementation below, M×M represents the size of the image. In other words, we have taken the value of M to be 128, 64, 32, 16 etc.

4.4.2 Choice of Basis functions :

The decomposition and reconstruction of the above image have been studied for three special basis functions, namely the Haar basis, the Linear Spline basis and the Cubic spline basis. The quality of reconstruction for all the three bases are compared to see as to which one of the three would give the best decomposition and the subsequent reconstruction. The mathematical formulation of the various basis are as follows [for calculation of the filter coefficients, see Appendix] :

4.4.2.1 Haar Basis

For the Haar basis function, h(n)'s has been calculated and tabulated as follows :

n	h(n)
0	0.5
1	0.5

4.4.2.2 Linear Spline

For the Linear Spline basis function, h(n)'s has been calculated and tabulated as follows :

n	h(n)
0	0.25
1	0.416
2	0.25
3	0.041
-1	0.041

4.4.2.3 Cubic Spline

For the Cubic Spline basis function, h(n)'s has been calculated and tabulated as follows :

n	h(n)	n	h(n)
0	0.542	6	0.012
1	0.307	7	-0.013
2	-0.035	8	0.006
3	-0.078	9	0.006
4	0.023	10	-0.003
5	-0.030	11	-0.002

The transfer function of the filters H, G, $\tilde{H}$ and $\tilde{G}$ corresponding to all the above three basis functions have been calculated on the basis of formulas given earlier.

4.4.3 Addition Of Noise in the Original Image

Some noise has been added to the original image. Then the decomposition and reconstruction of this corrupted image has been done in order to investigate whether such a procedure *using these basis* results in an improvement of the signal–to–noise ratio (SNR) or not. The various types of noise which have been included are :–

(a) Gaussian Noise with various means and variances.

and outputs (9–N) rows to the processor (N–1). Hence at the root processor, we once again get back 8 rows each containing M columns.

Step 7. The above procedure (from Step 3) is repeated for each of the M/8 parts.

Step 8. All the resultant parts are combined to get a data array of size (M×M/2).

Step 9. The data array is transposed in the root processor so that the rows become columns and vice–versa.

Step 10. The array is partitioned into M/16 parts, each having 8 rows.

Step 11. The above procedure (from Step 3) is repeated again.

Therefore, finally we get a data array of size (M/2 × M/2). Depending upon the filters used in the two steps, the final output is either the coarse signal or any of the three detail signals. The coarse signal $A^d_2 f$ can then be further decomposed by using a similar procedure. We thus get the wavelet representation of the original image upto whatever resolution we please.

4.4.4 Implementation of Reconstruction Algorithm

Step 1. The four signals viz., $A^d_{2^{j+1}}$, $D^1_{2^j} f$, $D^2_{2^j} f$, $D^3_{2^j} f$ (say, each of size M/2×M/2) are taken into the root processor.

Step 2. Alternate columns of zero are introduced in each of the data arrays so that now the size of each array becomes M/2 × M.

Step 3. All the four arrays are partitioned into M/8 parts of 4 rows each.

Step 4. The first part is taken and the first 4 rows of each array is inputted to the processor No. 1. The processor No. 1 accepts all the rows but keeps just two rows : one of $A^d_{2^j} f$ and the other of $D^1_{2^j} f$. It lets the other rows pass through to the next processor. In a similar manner,

the first four processors keep 2 rows, one concerning $A^{d}_{2^j}f$ and the other concerning $D^{1}_{2^j}f$. The next four processors keep one row each of $D^{2}_{2^j}f$ and $D^{3}_{2^j}f$ respectively.

Step 5. The rows of $A^{d}_{2^j}f$, $D^{1}_{2^j}f$, $D^{2}_{2^j}f$, and $D^{3}_{2^j}f$ are convolved with the filters G, H, G, and H respectively in the concerned processors.

Step 6. The result of these convolutions are added in the concerned processors itself as shown in the Figure 2.2. So, at this stage we have one row each of M columns in each of the processors.

Step 7. The rows are fed back to the root processor starting from the end processor and passing through each of the in–between processors.

Step 8. The above procedure is repeated (from Step 4) for each of the M/8 parts so that at the end we are left with two sets, each of size M/2 × M at the root processor.

Step 9. We now introduce alternate rows of zeros in the two sets so that their new size become M × M.

Step 10. Both the sets are again partitioned into M/8 parts of 8 rows each.

Step 11. The first 8 rows of both the sets are transmitted to the transputer array. Each transputer keeps one row of both the sets and lets the other rows go to the next processor.

Step 12. In each of the processor, the row of the first set is convolved with the filter G while the row of the second set is convolved with the filter H. The resultant of the two convolutions in each processor are added.

Step 13. The rows are fed back to the root processor starting from the end–processor. They are combined together in the root processor.

Step 14. This process is repeated for each part (from Step 4) so that finally we have one data array of size M × M.

Step 15. Each data element is multiplied by 4 in order to get the signal $A^{d}_{2^{j+1}}f$.

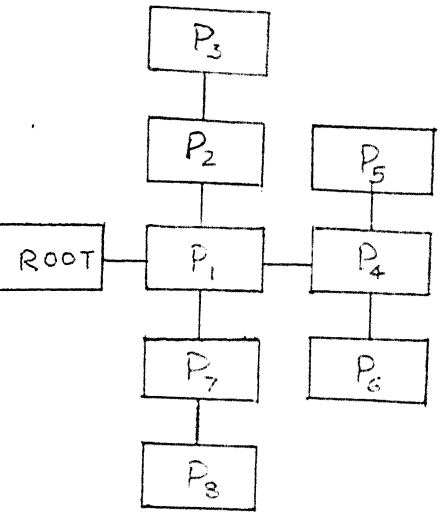


Fig. 4.1(a).

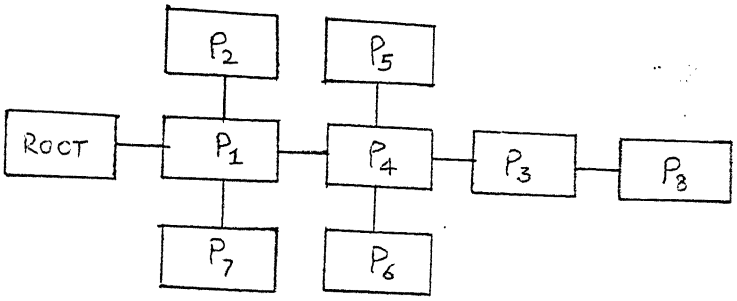


Fig. 4.1(b).

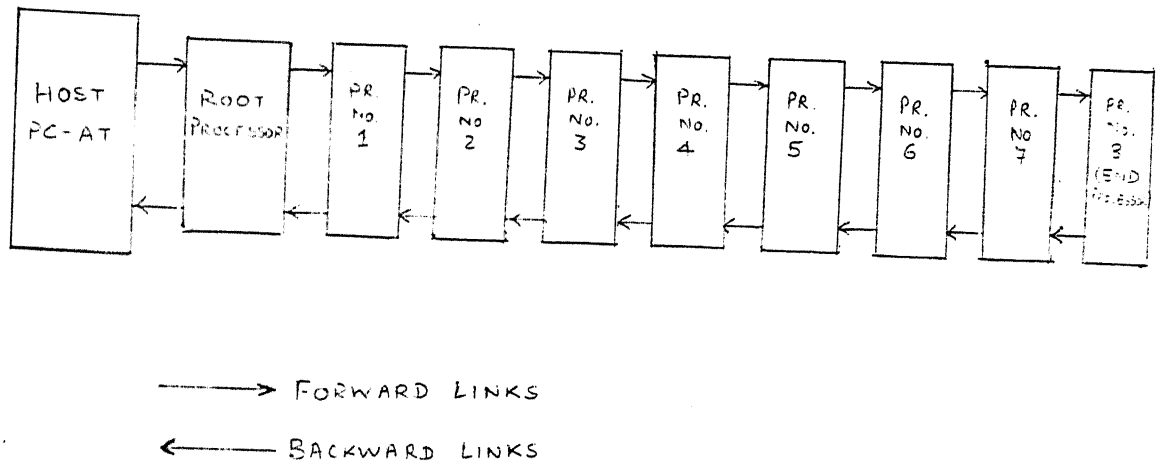


Fig. 4.1(c).

Fig. 4. Various Configurations of a eight node transputer network

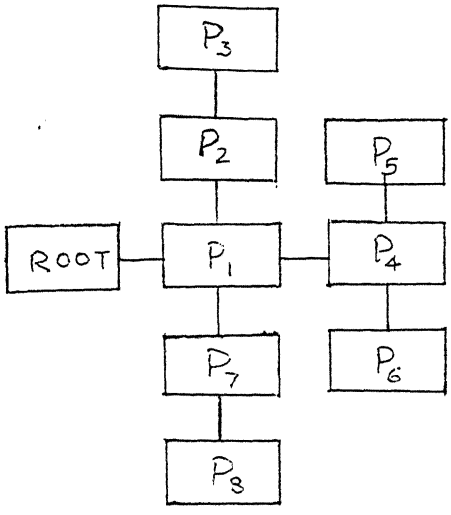


Fig. 4.1(a).

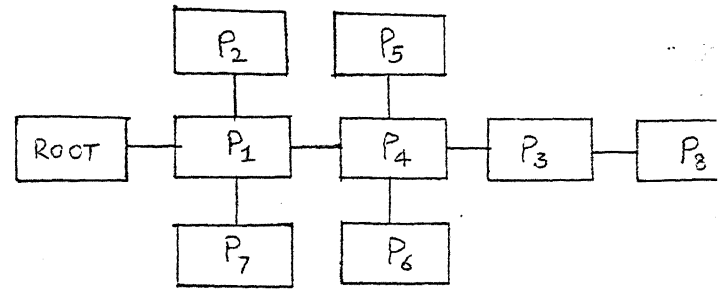


Fig. 4.1(b).

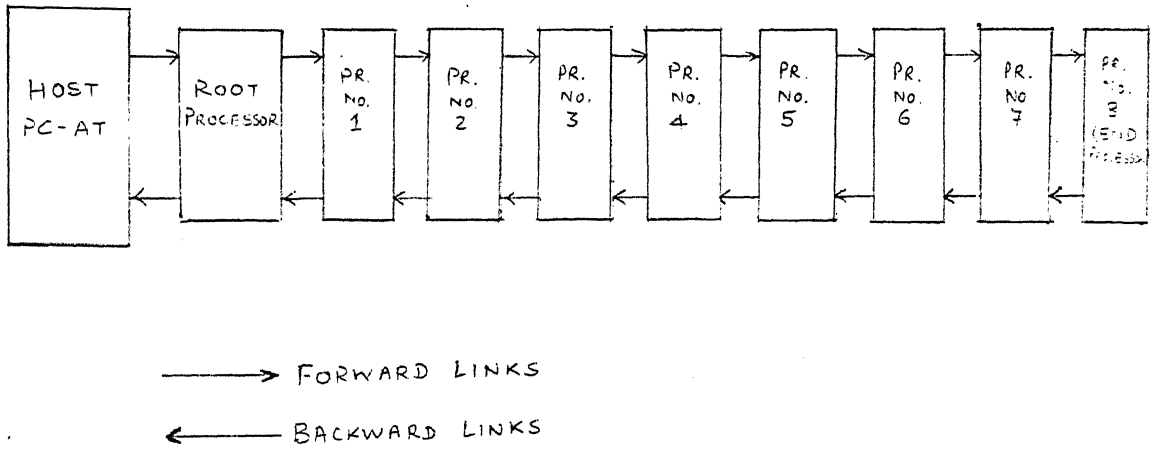
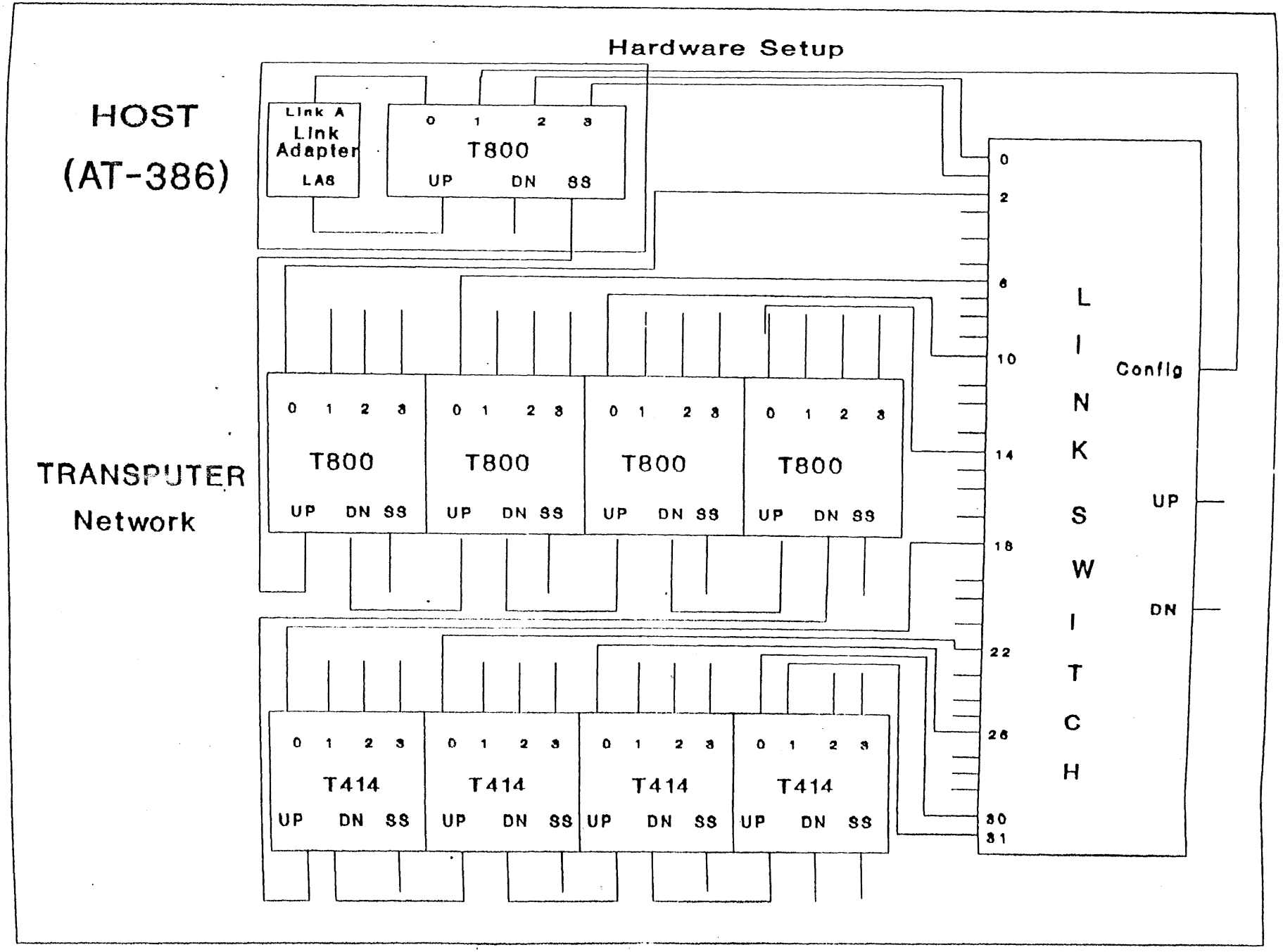


Fig. 4.1(c).

Fig. 4. Various Configurations of a eight node transputer network


Hardware Setup
HOST
(AT-386)
Link A
Link
Adapter
LA6
0 1 2 3
T800
UP DN SS
TRANSPUTER
Network
T800
T800
T800
T800
T414
T414
T414
T414
0
2
6
10
14
18
22
26
30
31
L
I
N
K
S
W
I
T
C
H
Config
UP
DN

CHAPTER 5

RESULTS AND CONCLUSIONS

5.1 RESULTS

The decomposition as well as reconstruction programs were run both on a single node transputer as well as the eight node transputer network. It was found that there was a dramatic decrease in the time taken for program execution when the eight node transputer network was used. On the average, the time taken by the eight node transputer network was $(1/6)^{th}$ of the time taken by the single transputer to run the same program.

For a (64×64) image, the time taken by the eight node transputer network for decomposition was 1.021 seconds on the average, with minor variations in case of different basis functions. The same image took 7.27 seconds when decomposed on a single node network. Similarly, an image of (128×128) size took 24.75 seconds and 141.36 seconds on the eight node transputer network and the single node transputer respectively.

For reconstructing an image of size (64×64), the eight node transputer network took 2.85 seconds on the average, while the single node transputer took 17.36 seconds on the average.

The decomposition and reconstruction of the images has been carried out with different basis functions viz., the Haar basis, linear spline and the cubic spline. The quality of reconstructed images was seen on the screen. However, to give a quantitative measure to the quality, it was measured in terms of the

normalized mean square error (nmse) which is defined as follows:–

$$\text{nmse} = [\ ((\text{Var}(o-r)\ /\ (\text{Var}(o)) \times 100\ ]\ \ \%$$

where Var(o–r) represents the variance of the original image minus the reconstructed image.

Var(i) for any image i is defined as $E((x-m)^2)$ where E is the expected value. m is the mean of the gray value of all the pixels and x is the gray value of any pixel.

From both the counts. i.e., through observation as well as through the criteria of normalized mean square error, it was found that the reconstruction is best in case of cubic spline and worst in case of Haar basis. The linear spline quality comes somewhere in between. The normalized mean square error (nmse) for the various cases are given below :

Case 1. Between the original image and the reconstructed image using Haar basis function, nmse = 14.12 %

Case 2. Between the original image and the reconstructed image using linear spline basis function, nmse = 10.78 %

Case 3. Between the original image and the reconstructed image using cubic spline basis function, nmse = 8.82 %

Case 4. Between the original image corrupted by Gaussian noise of mean 0.0 and standard deviation 2.0 and the reconstructed image using Haar basis function, nmse = 15.65 %

Case 5. Between the original image corrupted by Gaussian noise of mean 0.0 and and standard deviation 2.0 and the reconstructed image using linear spline basis function, nmse = 12.47 %

Case 6. Between the original image corrupted by Gaussian noise of mean 0.0 and and standard deviation 2.0 and the reconstructed image using cubic spline basis function, nmse = 10.63 %

Case 7. Between the original image corrupted by Gaussian noise of mean 0.0 and standard deviation 1.0 and the reconstructed image using Haar basis function, nmse = 14.96 %

Case 8. Between the original image corrupted by Gaussian noise of mean 0.0 and and standard deviation 1.0 and the reconstructed image using linear spline basis function, nmse = 11.68 %

Case 9. Between the original image corrupted by Gaussian noise of mean 0.0 and and standard deviation 1.0 and the reconstructed image using cubic spline basis function, nmse = 9.75 %

Case 10. Between the original image corrupted by Impulsive noise of magnitude 125.0 and the reconstructed image using Haar basis function, nmse = 15.89 %

Case 11. Between the original image corrupted by Impulsive noise of magnitude 125.0 and the reconstructed image using linear spline basis function, nmse = 13.38 %

Case 12. Between the original image corrupted by Impulsive noise of magnitude 125.0 and the reconstructed image using cubic spline basis function, nmse = 11.98 %

5.2 CONCLUSIONS

This work has described a mathematical model for the computation and interpretation of the concept of a multiresolution representation. It has also described a way of implementing this mathematical concept with the help of transputer networks. We explained how to extract the difference of information between successive resolutions and thus define a wavelet representation. This representation is computed by decomposing the original signal using a wavelet orthonormal basis, and can be interpreted as a decomposition using a set of independent frequency channels having a spatial orientation tuning. A wavelet representation lies between the spatial and Fourier domains. There is no redundant information because the wavelet functions are orthogonal. The computation is efficient due to the existence of a pyramidal algorithms based on convolutions with quadrature mirror filters. The original signal can be reconstructed from the wavelet decomposition with a similar algorithm. As far as wavelet decomposition and reconstruction of degraded image is concerned, there is no evidence that there is any noise reduction in the process for the types of noise and basis functions considered. This is, of course, no conclusive negation of the fact that there may be a noise reduction when multiresolution processing is done. It calls for further investigation.

5.3 SUGGESTIONS FOR FURTHER WORK

In this work, the wavelet decomposition and reconstruction has been implemented on a linear array of transputers. Hence, it can be investigated whether any other configuration of transputers gives enhanced performance or not. In real time applications, it is absolutely essential to reduce the time taken in processing the image. Several other configurations of the transputers are possible which might reduce the time taken and it would be worthwhile looking into them.

Further, the choice of basis functions in this work has been arbitrary. It would be worthwhile to use other basis functions for forming the wavelet representation. Moreover, the best basis functions can also be found out depending on the quality of the reconstructed image.

In this thesis, we investigated the effect of the three types of basis functions on images degraded by the Gaussian and impulsive noise. Our objective was to find out whether there is any noise reduction in the reconstructed image. However, for the types of noise and the types of basis functions used, there was no evidence of noise reduction. It would be interesting to find out whether any other type of noise or any other type of basis functions can lead to noise reduction.

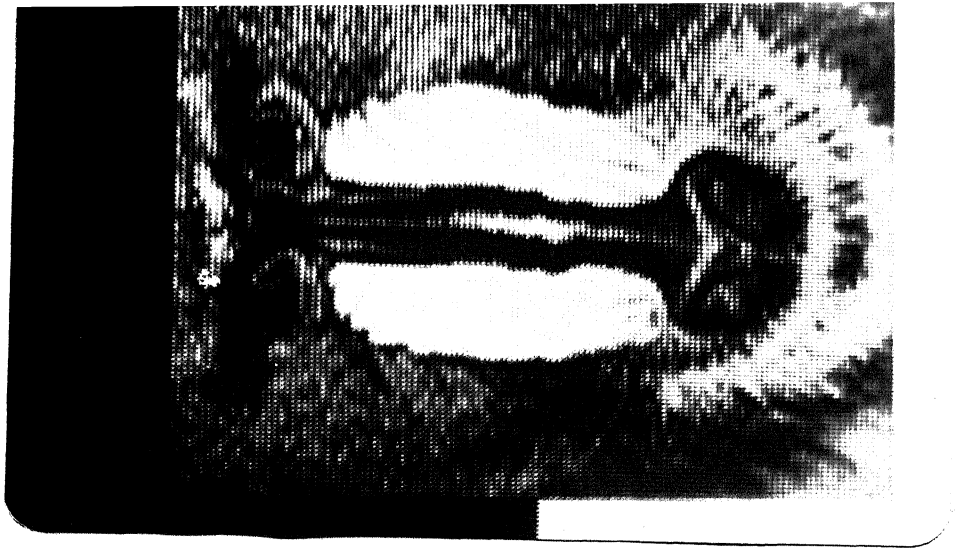


Plate 1. The Original Baboon image

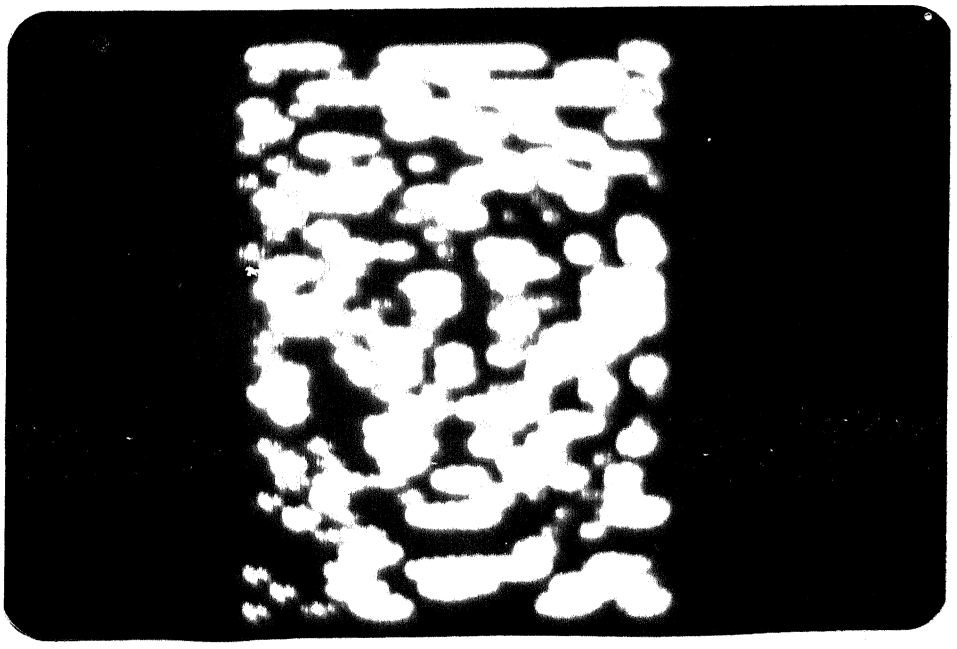


Plate 2. $D_{2^j}^1 f$, the first detail image using Haar basis function.

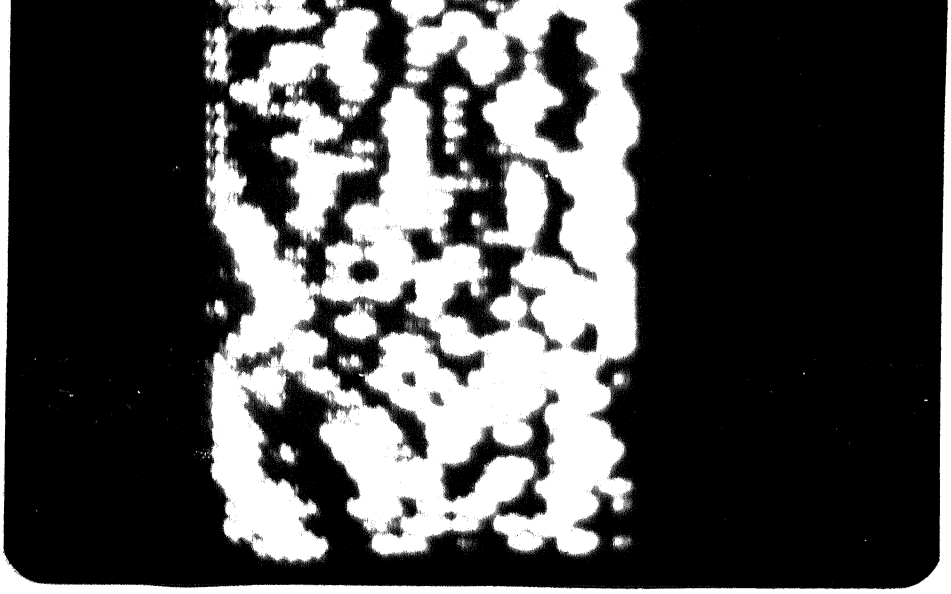


Plate 3. $D^2_{2^j}f$, the second detail image using Haar basis function.

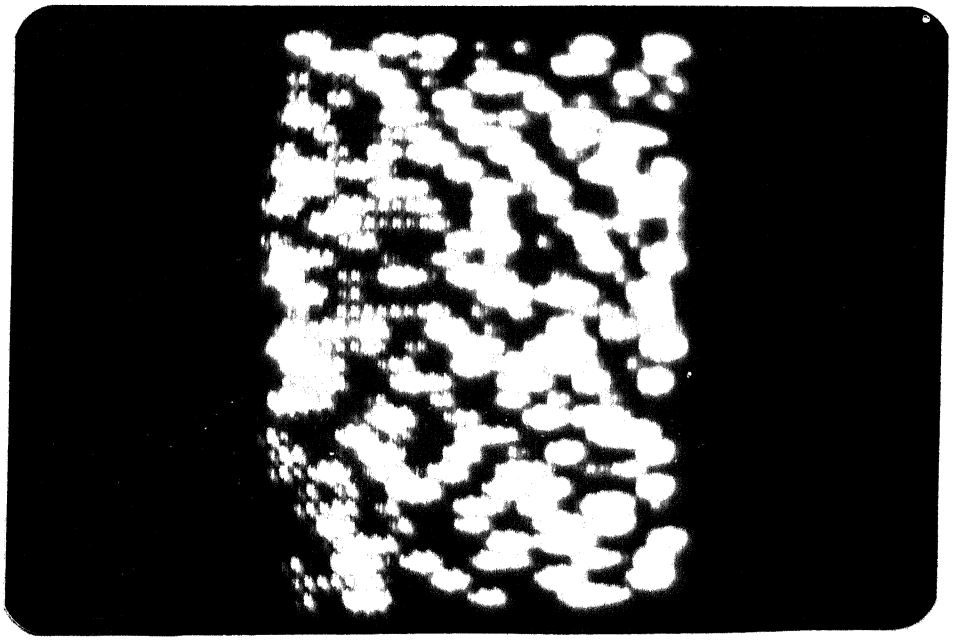


Plate 4. $D^3_{2^j}f$, the third detail image using Haar basis function.

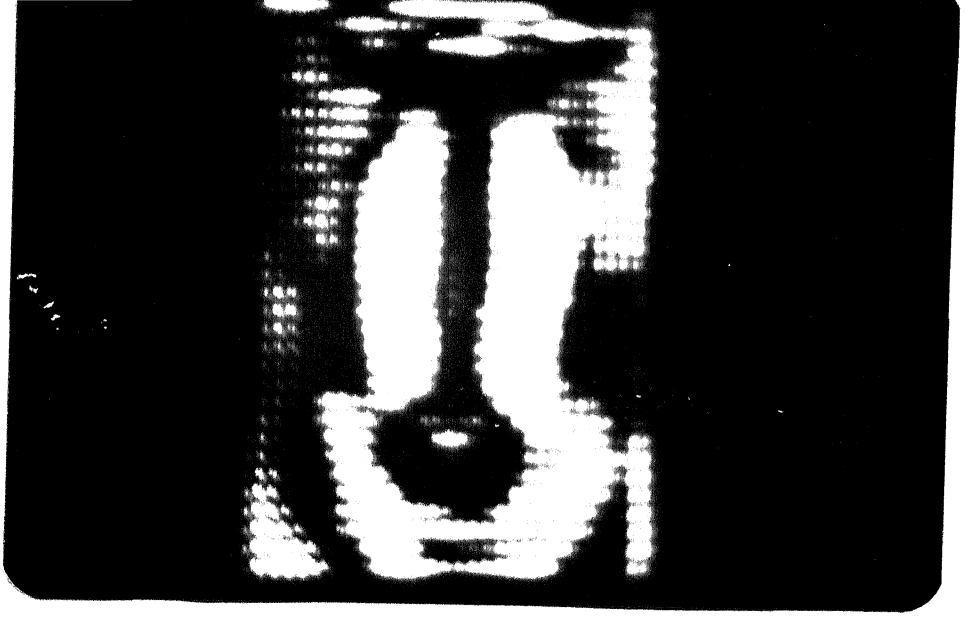


Plate 5. $A^{d}_{2^j}f$, the coarse image using Haar basis function.

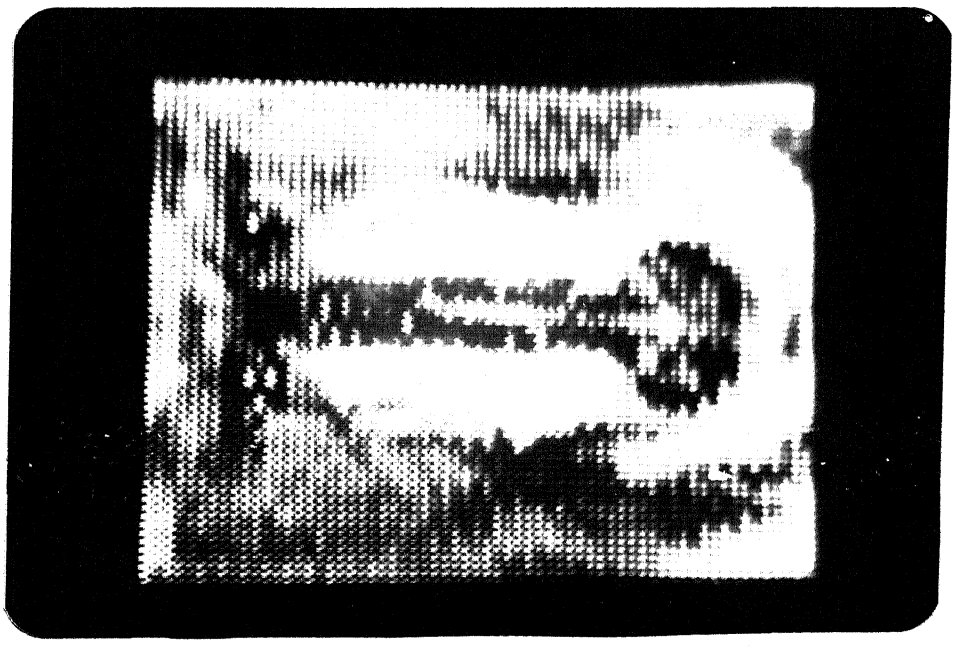


Plate 6. The Reconstructed image using Haar basis function.

Plate 7. The Reconstructed image using Linear Spline basis function.

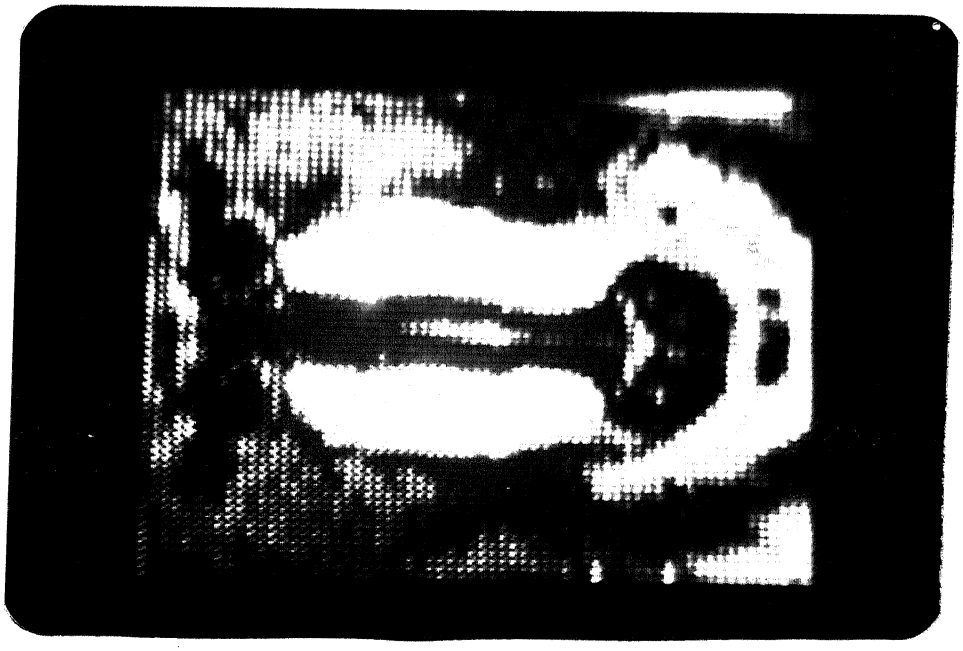


Plate 8. The Reconstructed image using Cubic Spline basis function.

Plate 9. The Original image degraded by Gaussian noise.

Plate 10.Reconstruction of the image corrupted by Gaussian noise

Plate 11. Reconstruction of the image corrupted by Gaussian noise using Linear Spline basis function.

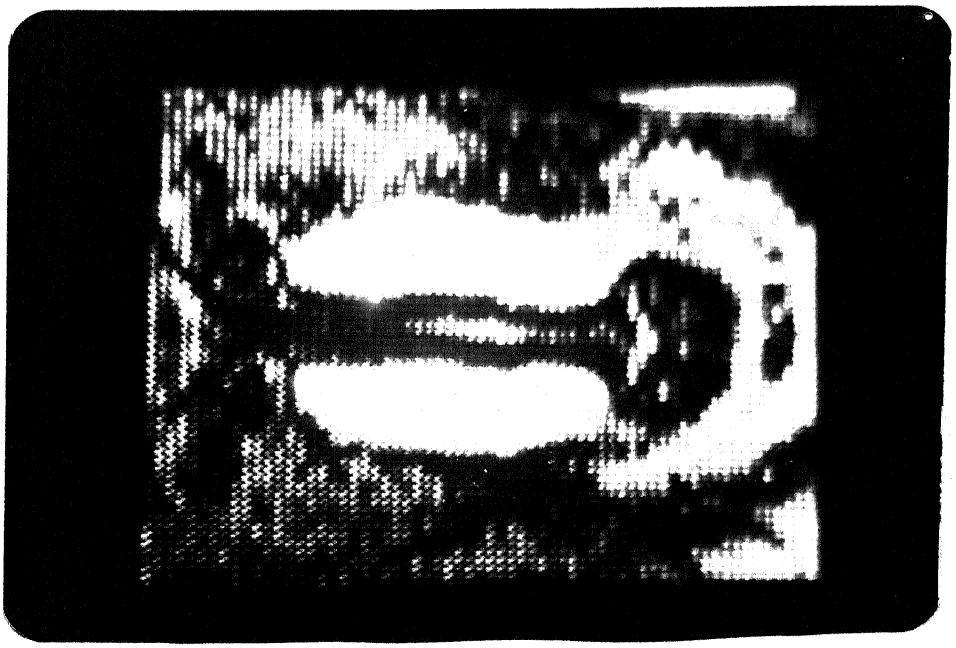


Plate 12. Reconstruction of the image corrupted by Gaussian noise using

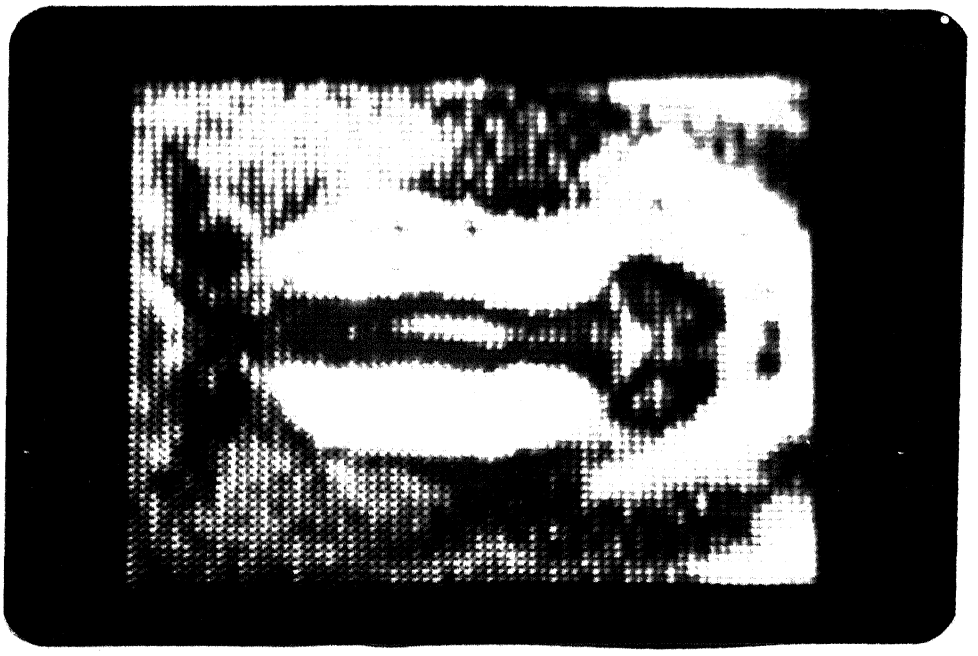


Plate 13. The Original image degraded by Impulsive noise.

Plate 14. Reconstruction of the image corrupted by Impulsive noise using

APPENDIX

DESIGN OF WAVELETS

The wavelet functions $\psi(x)$ have been constructed from orthogonal basis functions. Amongst the known orthogonal basis functions are Linear Spline, Haar basis, Quadratic Spline, and Cubic Spline.

We start with a function ϕ such that ϕ_{on} are an orthonormal basis for V_0. Since $\phi \in V_0 \subset \text{span} \left\{\phi(2.-n)\right\}$ there exists C_n such that

$$\phi(x) = \sum_n C_n \phi\ (2x-n)$$

Define then

$$\psi(x) = \sum_n (-1)^n\ C_{n+1}\ \phi(2x+n)$$

Then $\left\{\psi_{mn},\ m,n \in Z\right\}$ constitute an orthonormal basis of wavelets for $L^2(R)$.

Now we take up orthogonal basis functions one by one.

Haar Basis

The orthogonal function is given by

$$\phi(x) = \begin{cases} 1 & 0 \le x < 1 \\ 0 & \text{Otherwise} \end{cases}$$

$\phi(x)$ is also expressed as

$$\phi(x) = \phi(2x) + \phi(2x-1)$$

Since $\phi(x) = C_0\phi(2x) + C_1\phi(2x-1)$

Therefore,

$$C_0 = 1, \quad C_1 = 1.$$

By equation (3.2),

$$\psi(x) = \phi(2x) - \phi(2x-1)$$

Hence

$$\psi(x) = \begin{cases} 1 & 0 \le x \le 1 \\ -1 & 1/2 \le x < 1 \\ 0 & \text{Otherwise.} \end{cases}$$

2 Linear Spline

The function is given as

$$\phi(x) = \begin{cases} x & 0 \le x \le 1 \\ 2-x & 1 \le x \le 2 \\ 0 & \text{Otherwise.} \end{cases}$$

$\phi(x)$ is also expressed as

$$\phi(x) = \frac{1}{2}\phi(2x) + \phi(2x-1) + \frac{1}{2}\phi(2x-2).$$

Hence $C_0 = \frac{1}{2}$, $C_1 = 1$, $C_2 = \frac{1}{2}$

$$\psi(x) = -\frac{1}{2}\phi(2x+1) + \phi(2x) - \frac{1}{2}\phi(2x-1)$$

Hence

$$\begin{aligned}
\psi(x) &= -(x + \tfrac{1}{2}) && -\tfrac{1}{2} \le x \le 0 \\
&= (3x - \tfrac{1}{2}) && 0 \le x \le \tfrac{1}{2} \\
&= (\tfrac{3}{2} - x) && \tfrac{1}{2} \le x \le 1 \\
&= (\tfrac{3}{2} - x) && 1 \le x \le \tfrac{1}{2}
\end{aligned}$$

DESIGN OF FILTERS

h(n) is the impulse response of discrete filter H. It is expressed as

$$h(n) = \frac{1}{2}\int \phi\,(x/2).\phi\,(x-n)\,dx.$$

REFERENCES :

1) Stephane G. Mallat , A Theory for multiresolution signal decomposition : The Wavelet Representation., IEEE Transactions on Pattern Analysis and Machine intelligence, Vol.11, No.7, July 1989.

2) Olivier Rioul and Martin Vetterli , Wavelets and signal processing., IEEE Signal Processing Magazine, October 1991.

3) Stephane G. Mallat, Multi frequency Channel Decomposition and Wavelet models., IEEE Transactions on Acoustics Speech and Signal Processing, Vol.37 No.12, December 1989.

4) A.Grossman, R.Kronland–Martinet, and J.Morlet, "Reading and understanding continuous wavelet Transforms", Proc. Int. Conf. Marseille, France, Dec.1987

5) I.Daubechies, "Orthonormal basis of compactly supported wavelets", Comm. in pure and applied Math., Vol.41,No.7,1988.

6) I.Daubechies, "The Wavelet Transform, Time Frequency Localization and Signal Analysis", IEEE Transactions on Information Theory, Vol.36, No.5, Sept.1990

7) R.E.Crochiere and L.R.Rabiner, Multirate Digital Signal Processing, Prentice–Hall, Englewood–Cliffs, NJ, 1983

8) IEEE Transactions on Information Theory, Special issue on Wavelet Transforms and multiresolution signal analysis., Jan.1992

9) A.Rosenfeld and M.Thurston, "Edge and curve detection for visual scene analysis,"IEEE Trans. Comput., Vol.C–20, 1971

11) A.Rosenfeld and M.Thurston, "Coarse–fine Template matching," IEEE Trans. Syst., Man., Cyber . Vol.SMC–7, 1977

13) P.J.Burt and E.H.Adelson, "The Laplacian Pyramid as a compact image code," IEEE Trans. Commun., Vol. COM–31, Apr.1983

14) Dick Pountain and David May, A Tutorial introduction to Occam programming., BSP Professional Books ,U.K. 1987.

15) INMOS Ltd. Transputer Development System Manual.

16) INMOS Ltd. Occam Reference Manual.

EE-1992-M-SRI-WAV